

Wybrane Zagadnienia Kryptograficzne

Notatki z wykładu:

Wykład 1 – Wprowadzenie

Bezpieczny system komputerowy:

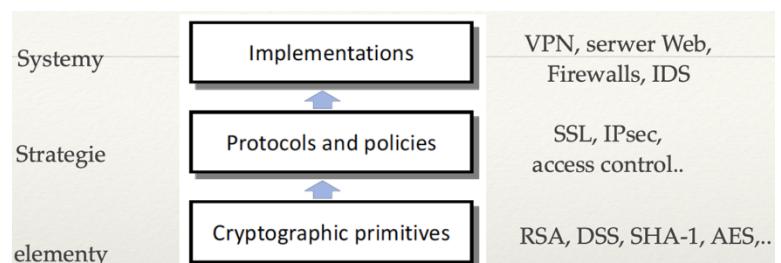
- Użytkownik może na nim polegać
- Zainstalowane oprogramowanie działa zgodnie ze swoją specyfikacją
- Wprowadzone na stałe dane nie zostaną utracone
- Dane nie ulegną zniekształceniu i nie zostaną pozyskane przez nikogo nieuprawnionego
- System jest bezpieczny tak długo jak długo nie istnieje skuteczna metoda ataku na niego (zazwyczaj możemy udowodnić, że jest odporny na poszczególne ataki ale nie oznacza to, że nie istnieje skuteczny atak!)

System wiarygodny:

- Dyspozycyjny (**available**) = dostępny na bieżąco
- Niezawodny (**reliable**) = odporny na awarie
- Bezpieczny (**secure**) = zapewniający ochronę danych
- Bezpieczny (**safe**) = bezpieczny dla otoczenia, przyjazny dla środowiska

Elementy bezpieczeństwa i ich wzajemne oddziaływanie:

- mechanizmy obrony na wszystkich poziomach muszą być „bezpieczne”
- muszą wzajemnie poprawnie oddziaływać - a jest to trudne zadanie



W rzeczywistości:

- Bezpieczeństwo nie zawsze jest rozważane na pierwszym miejscu:
- wydajność, użyteczność, przyjazność, koszty - mają pierwszeństwo
- Skomplikowane, nowoczesne systemy często nie są do końca zrozumiałe
 - złe założenia przyjęte dla protokołów na wyższych poziomach
- Błędne implementacje i konfiguracje
 - trudne do uniknięcia różnego podatności: przepełnienie bufora,..
- Sieci są dostępne i otwarte bardziej niż kiedykolwiek
 - zwiększa się ekspozycja, trudno kontrolować wszystkie ścieżki
- wiele z ataków nawet nie ma natury technicznej - inżynieria społeczna, ...

Kryteria oceny systemów informatycznych - cel bezpieczeństwa: CIA:

- **Confidentiality - Poufność** – ochrona przed ujawnieniem informacji:
 - przed nieautoryzowanym dostępem
 - dla przechowywanej i przesyłanej informacji
- **Integrity - Integralność** – ochrona przed modyfikacją
 - modyfikować dane mogą jedynie autoryzowane osoby lub procesy

- **Availability - Dostępność** – gwarancja uprawnionego dostępu
 - dostępne dane dla autoryzowanych osób, systemów
- Rozliczalność – określenie i weryfikacja odpowiedzialności
- Autentyczność – weryfikacja tożsamości
- Niezawodność – gwarancja odpowiedniego zachowania się systemu

Atak:

- wszelkie działanie, które narusza posiadane przez osobę lub organizację informacje czy dane
- Ataki dzielimy na:
 - Pasywne:
 - Np. przeczytanie zawartości dokumentu
 - Aktywne:
 - Np. blokowanie czyjegoś dostępu do danych
- często zagrożenie i atak oznacza to samo

Rodzaje ataków pod względem zakresu:

- **poufności** - nieautoryzowany dostęp:
 - Eavesdropping – podsłuch:
 - nieautoryzowany dostęp/przechwycenie danych
 - Traffic Analysis – analiza ruchu:
 - monitorując ruch danych można uzyskać jakąś informację nawet przy szyfrowanych danych
 - z analizy adresów, czasu, częstości itp - można wydedukować różne informacje
 - analizując kto wysyła coś do kogoś można wydedukować, gdzie adresat się znajduje
- **integralności** - modyfikacja lub zniszczenie:
 - Modification - modyfikacja
 - atakujący modyfikuje wiadomość dla swojej korzyści
 - wymazuje lub opóźnia wiadomość
 - Masquerading - inaczej spoofing - podszywanie się
 - Replaying - odtwarzanie
 - atakujący przechwytuje kopię wiadomości wysłanej przez nadawcę i może później próbować przesłać ją do odbiorcy
 - Repudiation - zaprzeczenie nadania lub odebrania (środki zabezpieczające są często nietechniczne - bilingi, logi, ..)
- **dostępności** - zakłócenie dostępności informacji lub systemu informatycznego:
 - Denial of Service
 - spowalnia albo uniemożliwia dostęp do systemu
 - atakujący może:
 - wysyłać fałszywe żądania do serwera (blokując w konsekwencji serwer)
 - przechwycić albo usunąć odpowiedź serwera do klienta blokować żądania klienta, tak że ten wysyła je wielokrotnie

Mechanizmy/usługi bezpieczeństwa:

- Mechanizmy bezpieczeństwa - mechanizmy zaprojektowane, żeby wykrywać, zapobiegać albo odtworzyć (przed /po ataku)
- Usługi bezpieczeństwa
 - usługi, które zapewniają bezpieczeństwo podczas przetwarzania i przesyłania danych
 - usługi wykorzystują jeden lub więcej mechanizmy bezpieczeństwa
- Najważniejsze
 - Szyfrowanie
 - Funkcja skrótu
 - Podpis elektroniczny
 - Uwierzytelnianie
 - Uzgadnianie klucza
 - Traffic padding - utrudnianie śledzenia ruchu
 - Routing control - stała zmiana dostępnych dróg dla pakietów uniemożliwiająca podsłuch Notaryzacja - wybór TTP - trzeciej strony zaufanej – certyfikaty

Polityka bezpieczeństwa:

- Zbiór reguł i praktyk, które określają jakie usługi bezpieczeństwa są używane do wrażliwych zasobów w systemie albo organizacji
- Zarządzanie bezpieczeństwem informacji powinno uwzględniać:
 - aktywa - dane, usługi w systemie, możliwości systemu - wydajność,.. wyposażenie systemu, składowe systemu (sprzęt, ludzie,...)
 - zagrożenia - określenie ich pomoże ustalić politykę bezpieczeństwa i mechanizmy i usługi jakie trzeba zaimplementować
 - trzeba zastanowić się skąd może pochodzić atak/niebezpieczeństwo - Kryminaliści, Włamywacze- pasjonaci, pracownicy, nieświadomienni pracownicy, służby, terroryści, ..

Wykład 2 – Kryptografia symetryczna:

Podział systemów kryptograficznych ze względu na:

- rodzaj klucza kryptograficznego:
 - Kryptografia symetryczna (z jednym kluczem tajnym):
 - Nadawca i odbiorca mają ten sam klucz
 - Kryptografia asymetryczna (jeden z kluczy jest tajny, drugi publiczny)
- Bezpieczeństwo:
 - Bezpieczeństwo bezwarunkowe:
 - niezależnie od ilości przechwyconego tekstu zaszyfrowanego, nie ma w nim wystarczająco dużo informacji, aby jednoznacznie określić tekst jawnego
 - Bezpieczeństwo obliczeniowe:
 - szyfru nie można złamać przy zastosowaniu systematycznej analizy z użyciem dostępnych zasobów

Kiedy szyfr jest bezpieczny:

- szyfr jest bezpieczny, jeśli nie istnieje atak, który w „rozsądnym czasie” jest w stanie go złamać.
- nie można wyznaczyć klucza na podstawie tekstów jawnych i szyfrogramów
- nie można wyznaczyć tekstu jawnego na podstawie szyfrogramu
- zależność pomiędzy tekstem jawnym i szyfrogramem powinna być nieliniowa

Typy ataków kryptoanalytycznych (założenie: atakujący zna: szyfrogram i algorytm szyfrujący)

- znamy szyfrogram
 - chcemy poznać tekst jawnny
- znamy tekst jawnny
 - chcemy poznać kilka par tekstu/szyfrogramu
- znamy wybrany tekst jawnny
 - wybieramy tekst jawnego i odpowiadający mu tekst zaszyfrowany
- znamy wybrany tekst zaszyfrowany
 - wybieramy szyfrogram i odpowiadający mu tekst jawnego
- znamy wybrany tekst
- wybieramy albo tekst jawnego albo zaszyfrowany do zaszyfrowania lub odszyfrowania aby przeprowadzić atak na szyfr

Założenie: wygenerowanie 1 ciągu zajmuje $10^{-6}s$

Atak poprzez wyczerpujące przeszukiwanie (brute force search):

- brute force search
- zawsze możemy przeszukać wszystkie możliwe klucze
- najbardziej podstawowy rodzaj ataku, proporcjonalny do rozmiaru klucza

n	Generowanie wszystkich ciągów [s]		Przyspieszenie poprzez zastosowanie komputerów wieloprocesorowych i programowania współbieżnego [$\times 10^{-9}$]
40	$1,1 \cdot 10^6$	=12,72 dnia	=0,001 s
64	$1,8 \cdot 10^{12}$	=7 miesięcy	=5,12 dnia
128	$3,4 \cdot 10^{22}$	= $1,08 \cdot 10^{25}$ lat	= $1,08 \cdot 10^{16}$ lat

Wiek Ziemi : $3,6 \cdot 10^9$ lat!

- na tekst częściowo znany, ze znanym tekstem jawnym
- z wybranym tekstem jawnym

Zasada Kerckoffsa:

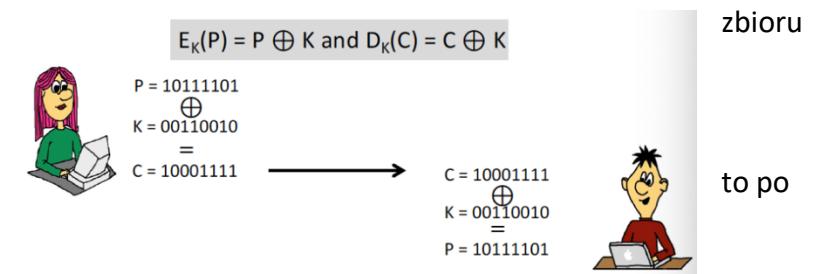
- Kryptosystem powinien być bezpieczny nawet, jeśli cały mechanizm szyfrowania (zasada działania szyfru), oprócz wartości tajnego klucza, jest znany nieprzyjaznemu kryptoanalitykowi.
- Inaczej niż bezpieczeństwo zapewniane przez ukrycie - security through/by obscurity
- Przykład:
 - Jeśli wezmę list, zamknę go w sejfie, ukryję sejf gdzieś w Nowym Jorku i każę ci go przeczytać, to nie jest bezpieczeństwo. To niejawność.
 - Z drugiej strony, jeśli wezmę list, zamknę go w sejfie, a następnie przekażę ci ten sejf wraz z jego specyfikacją techniczną oraz setką identycznych sejfów z ich kodami, abyś razem z najlepszymi włamywaczami świata mógł przestudiować jego zabezpieczenia – a Ty nadal nie będziesz go mógł otworzyć i przeczytać listu – to właśnie jest bezpieczeństwo.

Teoria Shannona i idealny sekret:

- P – wiadomość plaintext, C – zaszyfrowany text, K – klucz, E_K – enkrypcja kluczem K.
- Niech $|P| = |C| = |K|$ oraz $\Pr(P) > 0$ dla każdego „plaintext” P. Wtedy:
- Schemat szyfrowania daje idealną tajność wtedy i tylko wtedy, jeżeli każde K jest równie prawdopodobne oraz dla każdego P należącego do P oraz C należącego do C jest tylko jedno K należące do K z $E_K(P) = C$

Szyfr One-Time Pad (OTP):

- $P = C = K = \{0,1\}^n$, dla n ze N
- Dla zaszyfrowania P wybierz losowo nowy klucz K
- Szyfrowanie i deszyfrowanie prostu operacja XOR za pomocą klucza.



Szyfry klasyczne:

- 2 podstawowe techniki:
 - Przetwarzanie (transpositions/permuations):
 - szyfry permutacyjne
 - ukrywają wiadomość przestawiając porządek liter
 - bez zmiany alfabetu wiadomości
 - szyfrogram ma dokładnie ten sam rozkład częstotliwości co tekst jawnny
 - Podstawianie (substitutions):
 - Szyfry proste (monoalfabetyczne):
 - Szyfr addytywny (Cezara)
 - Mnożyciwy
 - Afiniczny
- prosty szyfr podstawieniowy zastępuje każdy znak alfabetu jawnego A odpowiadającym mu znakiem alfabetu tajnego B:
 - Jeżeli A = {a₁, a₂, ..., a_n}, B = {f(a₁), f(a₂), ..., f(a_n)}, to:
 - dla każdego znaku mi wiadomości m = m₁m₂... stosujemy przekształcenie E(m, f) = f(m₁), f(m₂), ..., f(m_n)

Szyfry podstawieniowe:

Szyfr Cezara:

- pierwszy znany szyfr podstawieniowy
- użyty i opisany przez Juliusza Cezara w wojnie Galickiej zastępował każdą literę literą przesuniętą o 3 pozycje w alfabetie
- przykład:
 - meet me after the toga party
 - PHHW PH DIWHU WKH WRJD SDUWB
- Przekształcenie możemy zdefiniować jako:
 - Abcdefghijklmnopqrstuvwxyz -> DEFGHIJKLMNOPQRSTUVWXYZABC
- Matematycznie możemy przypisać każdej literze liczbę:
 - Abcdefghijklmnopqrstuvwxyz -> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
- Szyfr Cezara miałby postać:
 - $C = E(p) = (p + k) \text{ mod } (26)$, $p = D(C) = (C - k) \text{ mod } (26)$
- kryptoanaliza szyfru Cezara:
 - tylko 26 możliwych kluczy (25) -> można wszystkie przeanalizować
 - mając szyfrogram, należy wypróbować wszystkie możliwe przesunięcia

Szyfry monoalfabetyczne:

- podstawienie, zamiast zwykłego przesunięcia
- każda litera zastąpiona zostaje różną losową literą alfabetu tajnego
- klucz nadal ma długość 26
- Przykład:
 - Plain: abcdefghijklmnopqrstuvwxyz -> Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN
 - Plaintext: ifwewishtoreplaceletters -> Ciphertext: WIRFRWAJUHYFTSDVFSUUUFYA
- Analiza częstotliwościowa:
 - litery są używane z różną częstością
 - w angielskim e jest najczęstszą literą, potem t,r,n,i,o,a,s
 - istnieją tabele częstotliwościowe dla pojedynczych liter, par i trójek
- Użyteczność tabel częstości liter w kryptoanalizie:
 - podstawienie monoalfabetyczne nie zmienia częstości występowania liter
 - oblicza się częstotliwości dla szyfrogramu -> porównuje z tabelami dla odpowiedniego alfabetu
 - pomocne są tabele dla par, trójek dla identyfikacji alfabetu
 - Przykład:
 - Dany tekst zaszyfrowany:
 - UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZVUEPHZHMDZ SHZOWSFAPPDTSPQUZWYMXUZUHSXEPMYEPOPDZSZUFPOMBZWPFPUPZHM DJUDTMOMHQ
 - Na podstawie obliczonej częstości występowania liter Zamiana P & Z na e & t, dalej ZW na th, a ZWP na the
 - Ostatecznie otrzymamy:
 - it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the vietcong in Moscow

Szyfr Playfair:

- szyfrowanie tekstu na przykład parami liter
- dana jest macierz 5x5 jako klucz szyfru wypełniona 25 literami bez powtórzeń (i=j w tym systemie)
- przykład:
 - 1 słowo kluczowe =MONARCHY a potem wszystkie kolejne litery alfabetu bez tych w słowie kluczowym:
MONAR
CHYBD
EFGIK
LPQST
UVWXZ
- Szyfrowanie:
 - Jeżeli w parze liter występuje powtórzenie, wstawiamy filtr (np.: „X”):
 - „balloon” -> „ba lx lo on”
 - Jeżeli obie litery wypadają w tym samym rzędzie, to zastępujemy każdą literę literą z prawej:
 - „ar” -> „RM”
 - Jeżeli obie litery występują w tej samej kolumnie, zastępujemy każdą z liter literą z dołu:
 - „mu” -> „CM”
 - W przypadku różnego wiersza i kolumny, zastępujemy literami tworzącymi prostokąt z literami szyfrowanymi:
 - „hs” -> „BP”, „ea” -> „IM”
- Bezpieczeństwo:
 - większa niż monoalfabetycznych!
 - $26 \times 26 = 676$ diagramów
 - potrzeba tablicy częstości występowania liter o 676-wejściach
 - równocześnie większą ilość szyfrogramu do stworzenia takiej tablicy
 - szeroko stosowany przez wiele lat np. w czasie I wojny światowej
 - może być złamany, mając kilkaset liter
 - liczba możliwych kluczy -> 25!

Szyfry polialfabetyczne:

- Używają wielu alfabetów szyfrujących
- utrudnia kryptoanalizę przez zastosowanie większej liczby alfabetów szyfrujących i spłaszcza rozkład częstotliwości
- używa się klucza, dla oznaczenia który z alfabetów jest stosowany dla której litery wiadomości
- w jednej rundzie użyty zostaje każdy z alfabetów
- powtarza się od początku do końca klucza

Szyfr Vigenere'a:

- stosuje wiele przesunięć cezara zależnych od klucza $E(m, k_i) = (m + k_i) \bmod n$,
- gdzie k jest kluczem okresowym zdefiniowanym jako $k = k_1 k_2 \dots k_T k_{T+1} \dots$
- deszyfrowanie jest operacją odwrotną
- Przykład:
 - mając dany tekst jawnny i klucz, każdą literę tekstu szyfrujemy przesunięciem cezara o długość przesunięcia litery na odpowiedniej pozycji klucza:
 - Key: deceptive
 - plaintext: wearediscoveredsaveyourself
 - ciphertext: ZICVTWQNGRZGVTVAVZHCQYGLMGJ
- Bezpieczeństwo:
 - zmieniona jest częstość występowania liter lecz nie do końca zatarta
 - kryptoanalizę rozpoczynamy od sprawdzenia rozkładu częstości a następnie staramy się określić okres klucza (liczbę alfabetów)

Metoda Kasiskiego:

- powtórzenia w szyfrogramie wskazują okresowość szyfru
- znajdujemy powtórzenia w szyfrogramie
- sugerowany rozmiar klucza 3 lub 9
- znajdujemy inne powtórzenia, uzgadniając wspólny okres
- atakujemy każdy z monoalfabetycznych szyfrów osobno
- Przykład: (Wybieramy VTW)
 - key: deceptive
 - plaintext: wearediscoveredsaveyourself
 - ciphertext: ZICVTWQNGRZGVTVAVZHCQYGLMGJ

Szyfry homofoniczne:

- szyfr podstawieniowy, w którym każdej literze tekstu jawnego odpowiada inny zbiór symboli kryptogramu (homofonów)

$f(\cup)$	$= \{Q, V\}$	$f(I)$	$= \{L, 6\}$	$f(R)$	$= \{U\}$
$f(A)$	$= \{H, 9, &\}$	$f(J)$	$= \{S\}$	$f(S)$	$= \{Z, 5\}$
$f(B)$	$= \{2, >\}$	$f(K)$	$= \{I, W\}$	$f(T)$	$= \{C\}$
$f(C)$	$= \{R\}$	$f(L)$	$= \{A\}$	$f(U)$	$= \{M, \#\}$
$f(D)$	$= \{E\}$	$f(M)$	$= \{Y, 3\}$	$f(V)$	$= \{G\}$
$f(E)$	$= \{_, X, ;\}$	$f(N)$	$= \{T, /\}$	$f(W)$	$= \{4, +\}$
$f(F)$	$= \{J, ~\}$	$f(O)$	$= \{D, 7, 8, =\}$	$f(X)$	$= \{F\}$
$f(G)$	$= \{B, ?\}$	$f(P)$	$= \{O\}$	$f(Y)$	$= \{!, \$\}$
$f(H)$	$= \{P, *, %\}$	$f(Q)$	$= \{K\}$	$f(Z)$	$= \{N, 1, @\}$

Szyfry przestawieniowe:

Szyfr półkowy:

- wpisujemy tekst np. w dwóch rzędach, po literze raz w jednym raz w drugim
- odczytujemy rzędami tekst zaszyfrowany
- meetmeafterthetogaparty -> m e m a t r h t g p r y
 e t e f e t e o a a t
- szyfrogram: MEMATRHTGPRYETEFETEOAAT

Szyfr transpozycji wierszy:

- wpisujemy tekst wierszami do odpowiedniej liczby kolumn
- zmieniamy porządek kolumn zgodnie z kluczem i odczytujemy szyfrogram kolumnami
- Key: 4 3 1 2 5 6 7
- Plaintext:
a t t a c k p
o s t p o n e
d u n t l l t
w o a m x y z
- Ciphertext: TTNAAPMTSUOAODWCOIXKNLYPETZ

Złożenia szyfrów:

- szyfry oparte na podstawieniach lub przestawieniach nie są bezpieczne z powodu charakterystycznych cech języka
- można rozpatrywać złożenia szyfrów, które zwiększą bezpieczeństwo, lecz
 - dwa podstawienia są bardziej złożonym podstawieniem
 - dwa przestawienia są bardziej złożonym przestawieniem
 - złożenie przestawienia i podstawienia znacznie zwiększa bezpieczeństwo szyfru!

Maszyny rotowe:

- wielostanowe algorytmy podstawieniowe
- najczęściej stosowane algorytmy przed współczesnymi algorytmami
- szeroko stosowane w czasie II wojny światowej – niemiecka enigma, japońska Purple
- implementacja złożonego, zmennego algorytmu podstawieniowego (liczne cylindry realizujące pojedyncze podstawienia, rotowane i zmieniane po każdej zaszyfrowanej literze)

Szyfry blokowe:

Szyfry blokowe vs strumieniowe:

- szyfratory blokowe dzielą wiadomość na bloki i dalej bloki te szyfrują (na bloki 64 i więcej bitowe)
- szyfratory strumieniowe przetwarzają wiadomość bit po bicie, lub bajtami wiele obecnych szyfrów stanowią szyfry blokowe

Założenia szyfratorów blokowych:

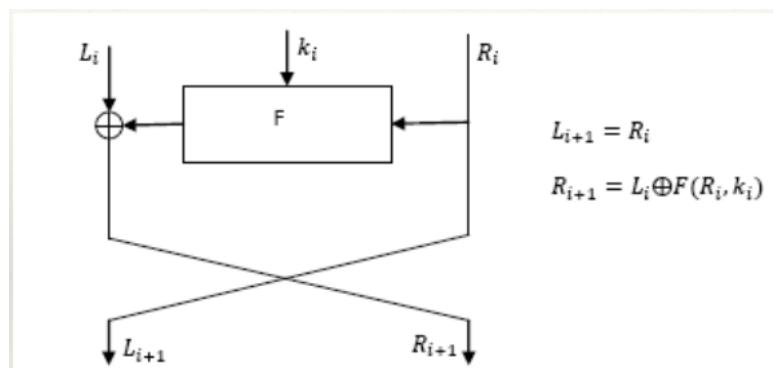
- szyfratory blokowe wyglądają jak ogromnie wielkie podstawienia
- potrzebowalibyśmy tablicę o 2^{64} wejściach dla bloków 64 bitowych
- wiele symetrycznych szyfratorów blokowych bazuje na przekształceniu Feistela

Sieci podstawieniowo-przestawieniowe (S-P) Shannona:

- Shannon: „lepiej składać proste algorytmy, które realizują mieszanie i rozpraszanie niż poszukiwać algorytmów złożonych”
- bazują na dwóch poznanych podstawowych operacjach szyfrujących:
 - **podstawieniach** (S-bloki) oraz **przestawieniach** (P-bloki)
- wprowadzają dyfuzję i konfuzję wiadomości:
 - **dyfuzja** - rozproszenie struktur statystycznych tekstu jawnego
 - **konfuzja** - komplikacja, maksymalne zatarcie związku pomiędzy szyfrogramem i kluczem

Struktura Feistela:

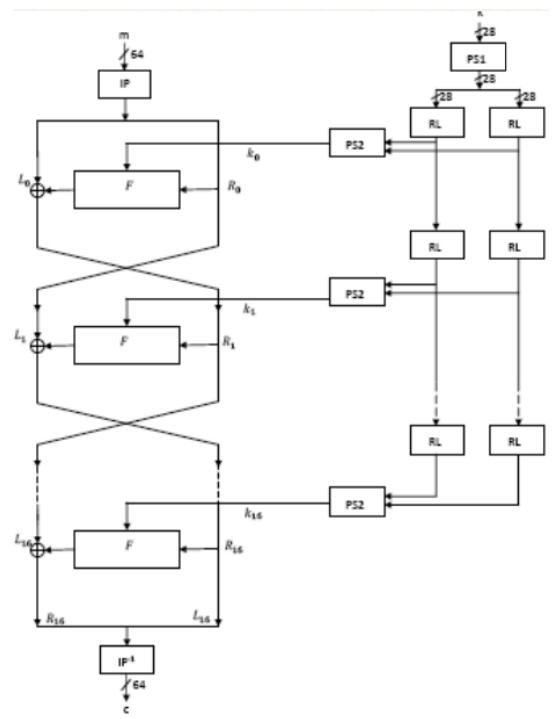
- n sekwencyjnych rund
- Podstawienie lewej połowy tekstu L_i
- Korki:
 - 1. Zastosowanie funkcji F dla prawej połowy wiadomości R_i
 - 2. operacja XOR wyjścia z 1. i L_i
- funkcję rundy charakteryzuje podklucz k_i (otrzymywany z ogólnego klucza K)
- Założenia projektowe Feistela:
 - rozmiar bloku:
 - większy rozmiar zwiększa bezpieczeństwo lecz spowalnia szyfr
 - rozmiar klucza:
 - większy rozmiar utrudnia zastosowanie wyczerpujące przeszukanie kluczy, spowalnia szyfr
 - liczba rund:
 - zwiększa bezpieczeństwo, spowalnia szyfr
 - podklucze i liczba rund
 - większa złożoność generacji zwiększa bezpieczeństwo, spowalnia szyfr
 - podobnie j.w.
 - szybka implementacja i łatwość analizy
- Aby odszyfrować tekst wystarczy jeszcze raz przepuścić go przez strukturę (z odwrotną kolejnością kluczy)



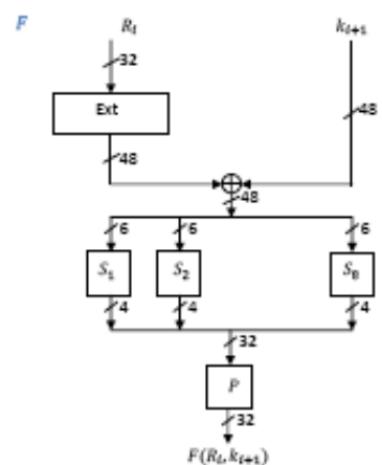
Algorytm DES:

DES:

- szyfruje wiadomości 64-bitowe używając 64-bitowego klucza, przy czym informacja użyteczna zajmuje 56 bitów (co ósmy bit to bit parzystości)
- Wykonywanych jest 16 cykli operacji nazywanych funkcjami f
- Powstanie klucza:
 - Ponieważ klucz jest 64-bitowy, redukowany jest do klucza 56 bitów przez pominięcie co ósmego bitu parzystości.
 - Tak przygotowany ciąg bitów poddawany jest permutacji wejściowej, po czym dzielony jest na dwa podciągi 28-bitowe (**PS1**).
 - Następnie połowy te przesuwane są w lewo (**RL**) o jeden lub dwa bity, zależnie od numeru cyklu (odpowiednia tabela przesunięć).
 - Po połączeniu nowopowstałych ciągów wybiera się 48 z 56 bitów (permutacja z kompresją – **PS2**).
 - Tak otrzymujemy klucz dla i-cyku (gdzie i jest numerem cyklu), $i = 1, \dots, 16$



- Realizacja funkcji f:
 - W funkcji f prawa połowa bloku danych jest poddawana permutacji z rozszerzeniem (**Ext**), czyli z 32 do 48 bitów.
 - Następnie, nowopowstały podciąg bitów, łączony jest za pomocą operacji XOR z 48 bitami przesuniętego i spermutowanego wcześniej klucza.
 - Po tej operacji otrzymany ciąg dzielony jest na 8 części i wprowadzany do skrzynek **S**-bloków, gdzie z 6-bitowych podciągów na wyjściu otrzymujemy 4-bitowe podciągi, które łączymy ze sobą, a następnie nowopowstały ciąg jest na wyjściu poddany permutacji (**P**) i otrzymujemy zaszyfrowany ciąg 32-bitowy.
- Następnie bity wyjściowe przekształconej prawej strony są dodawane do bitów lewej połowy danych.
- Zmieniona lewa połowa danych staje się prawą połową, natomiast poprzednia prawa połowa staje się nową lewą połową.
- Po wykonaniu wszystkich 16 powtórzeń funkcji Feistela, lewa i prawa połowa danych jest łączona za pomocą sumowania XOR.
- Na koniec przeprowadzana jest Permutacja Końcowa



Cechy DES:

- Nieliniowość (dla przekształceń liniowych i afinicznych można znaleźć macierz przekształcenia)
- Lawinowość - dla wszystkich możliwych kluczy, zmiana pojedynczego bitu wejścia powoduje zmianę w przybliżeniu połowy bitów na wyjściu
- SAC (ang. Strict Avalanche Criterion) - dla każdego klucza każdy bit wyjścia zmienia się z prawdopodobieństwem 0,5, gdy zmianie ulega jeden bit wejścia
- Zależność międzybitowa - dla wszystkich wartości klucza każdy bit wyjścia zależy od wszystkich bitów wejścia
- Dyfuzyjność - każdy bit wejścia sieci powinien w kolejnych iteracjach wpływać na coraz większą liczbę bitów.

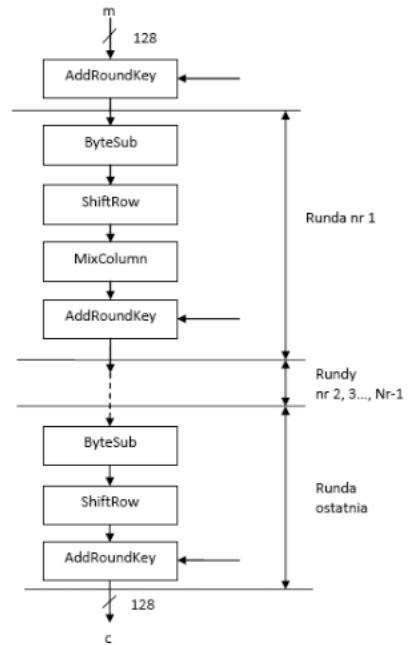
Problem z 2DES:

- użycie 2x DESa jednego za drugim, z niezależnymi kluczami K1 i K2
- Nie zwiększyło to efektywnej długości klucza
- Przykład - Attack in the middle:
 - założmy, że atakujący posiada parę: P i C, $\text{DES}_{K_2}(\text{DES}_{K_1}(P)) = C$
 - atakujący oblicza listę możliwych szyfrogramów Z dla P i K1, 2^{56} operacji
 - atakujący oblicza listę możliwych Z' dla C i K2, (maks 2^{56} operacji), tak dugo aż Z=Z'
mamy $2 * 2^{56}$ operacji aby odgadnąć K1 i K2 - czyli złożoność 2^{57}

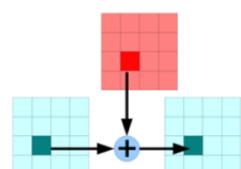
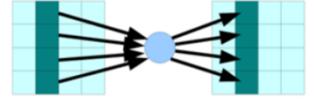
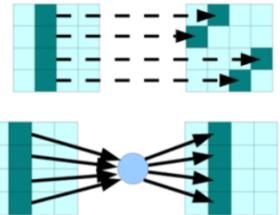
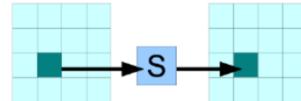
Algorytm AES:

AES:

- Długość przetwarzanego bloku: **128 bitów**
- długość klucza: **128 (10 rund), 192 (12 rund), 256 bitów (14 rund)**
- słowa wejściowe, wyjściowe i klucz są przetwarzane jako tablice bajtów
- Przekształcenia wykonywane w AES:
 - Podstawienia **ByteSub()**
 - Przesunięcia **ShiftRows()** wierszy w tablicy stanów
 - Mieszanie **MixColumns()** danych z każdą kolumną tablicy stanów
 - Dodawania **AddRoundKey()** klucza rundowego do tablicy stanu
- AES używa **Tablicy Stanów** (bajtową)



- Operacje używane w każdej rundzie:
 - Podstawienie bajtów (S-box - **ByteSub**):
 - Jedyny nieliniowy element szyfru AES
 - każdy element z tablicy stanów zastępowany jest elementem z S-bloku
 - Przesunięcie wierszy (**ShiftRows**):
 - Każdy wiersz, oprócz pierwszego jest cyklicznie w każdej rundzie przesuwany w lewo (odpowiednio o 1, 2 i 3 bajty)
 - Wymieszanie kolumn (**MixColumns**):
 - Mieszanie kolumn wykorzystuje mnożenie w ciele GF (2^8) modulo pierwiastek niereductowalny
 - Dodanie klucza (**KeyAddition**):
 - Do tablicy stanu dodawany jest 128 bitowy klucz rundowy - kazdy stan w tablicy jest xor-owany z odpowiednim bajtem klucza
- Ataki: brak praktycznych ataków przeciwko pełnorundowym algorytmom



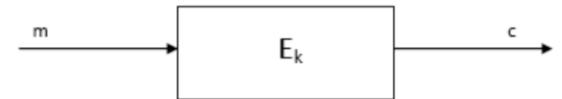
Tryby pracy szyfrów blokowych:

ECB – Electronic Code Book

- Każdy z bloków wiadomości jest kodowany oddzielnie. Podobnie, każdy blok szyfrogramu jest również deszyfrowany oddzielnie.
- Bezpieczne tylko dla $r=1$ (wiadomość nie dłuższa niż długość bloku)
- powtórzenia w wiadomości mogą być widoczne w szyfrogramie
- powtórzenia w kolejnych blokach wiadomości, szczególnie dane graficzne lub mało różniące się wiadomości, które stają się problemem analizy książki kodowej
- głównie stosowany do przesyłania niewielu bloków wiadomości

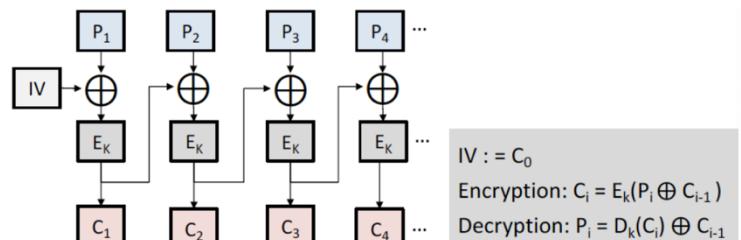
$$m = m_1 || m_2 || \dots || m_i || \dots || m_r$$

$||$ – konkatenacja, m_i – bloki wiadomości



CBC:

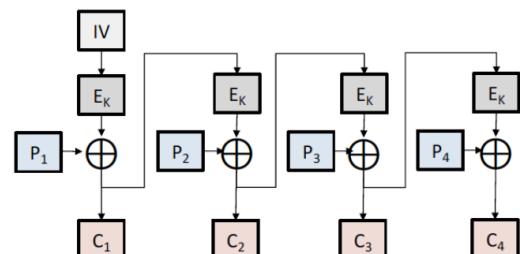
- Polega on na dodawaniu XOR każdego kolejnego bloku tekstu jawnego do poprzednio otrzymanego bloku szyfrogramu.
- Dopiero wynik tego działania jest szyfrowany w zwykły sposób.
- Pierwszy blok tekstu jawnego jest dodawany XOR do losowego wektora inicjującego (zwyczajowo oznaczanego jako IV), o takiej samej długości jak długość każdego bloku danych.
- Każdy kolejny blok szyfrogramu zależy więc od poprzedniego bloku szyfrogramu.
- Initial Value (IV) znana nadawcy i odbiorcy:
 - jeżeli IV jest przesyłana jawnie, atakujący może zmienić bity pierwszego bloku, kompensując potem zmianę IV
 - IV musi być zmienne albo musi zostać przesłane przed wiadomością, zaszyfrowane na przykład ECB



IV : = C_0
 Encryption: $C_i = E_k(P_i \oplus C_{i-1})$
 Decryption: $P_i = D_k(C_i) \oplus C_{i-1}$

CFB:

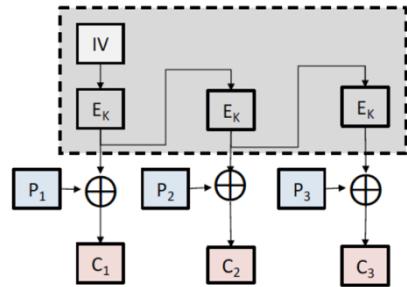
- Działa jak CBC, ale szyfruje się nie tyle bloki tekstu jawnego, lecz wymieszane dane z poprzedniej rundy, do których następnie dodaje się oryginalne bity wiadomości.
- Podczas deszyfrowania bloków szyfrogramu wykorzystuje się algorytmy szyfrujące, które zostały wcześniej użyte podczas szyfrowania.
- **Zakłamanie jednego bitu tekstu jawnego powoduje uszkodzenie aktualnego bloku szyfrogramu i wszystkich kolejnych bloków.**
- odpowiedni dla danych przychodzących jako bity lub bajty
- najczęściej stosowany szyfrator strumieniowy (używa się w kanałach nie narażonych na zakłócenia)
- szyfrator używany jest w trybie szyfrowania na obu końcach propagacja błędu



IV public, IV : = C_0
 Encryption: $C_i = E_k(C_{i-1} \oplus P_i)$
 Decryption: $P_i = C_i \oplus E_k(C_{i-1})$

OFB:

- Algorytmy szyfrujące działające w trybie OFB wytwarzają strumień bitów klucza, który potem wykorzystywany jest do kodowania kolejnych bloków danych. Pod tym względem, działanie szyfru blokowego upodabnia się do pracy typowego szyfru strumieniowego.
- Błędy nie są propagowane - uszkodzenie jednego bitu tekstu jawnego bądź szyfrogramu powoduje zakłamanie jednego, odpowiadającego mu, bitu odpowiednio szyfrogramu lub tekstu jawnego
- stosowany jest zatem w narażonym na zakłócenia środowisku albo kiedy szyfrujemy zanim wiadomość jest dostępna
- podobny do CFB, tyle, że sprzążone jest wyjście szyfratora i nie jest zależne od wiadomości
- potrzeba synchronizacji nadawcy i odbiorcy, potrzebna metoda ponownej synchronizacji, ponieważ „losowe” bity są niezależne od wiadomości, to nie można ich użyć więcej niż raz!



IV public
Encryption: $C_i = E_k^{i-1}(IV) \oplus P_i$
 $E_k^{i-1}(IV) = IV \text{ encrypted } i\text{-times}$
Decryption: $P_i = C_i \oplus E_k^{i-1}(IV)$

CTR:

- Użycie trybu CTR upodabnia działanie szyfru blokowego do szyfrów strumieniowych.
- Podobnie jak w przypadku trybu OFB, bity strumienia klucza tworzone są niezależnie od zawartości kolejno szyfrowanych bloków danych.
- W tym trybie szyfruje się kolejne wartości stale zwiększającego się licznika, zsumowane z dodatkową liczbą nazywaną nonce (nonce oznacza unikalny numer: number used once).
- Nonce pełni taką samą rolę jak wektor inicjujący w poprzednich trybach.
- Wydajność
 - stosowany do szyfrowania równoległego
 - w zależności od zapotrzebowania
 - dobry dla połączeń o dużej szybkości przepływu danych
- losowy dostęp do zaszyfrowanych bloków
 - nie potrzeba deszyfrować w kolejności od początku
- dowiedzione bezpieczeństwo (równie dobry jak inne tryby)
- musimy być pewni, że nie zostaną użyte ponownie wartości klucza/licznika, w przeciwnym wypadku może zostać złamany (podobnie jak OFB)

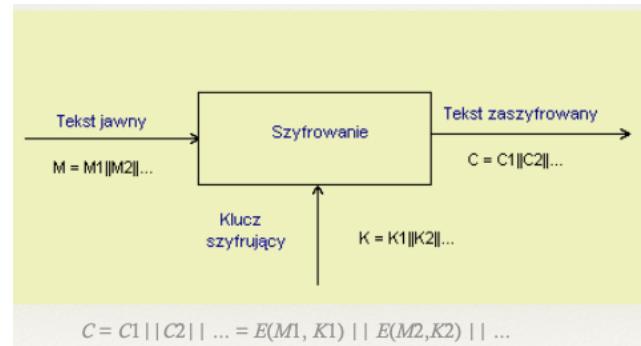
Dodatkowe informacje:

- ECB i CBC:
 - wymagają dopełnienia do pełnej długości bloków (Padding)
 - dopełnienie musi być łatwe do usunięcia
- Żaden z trybów nie chroni INTEGRALNOŚCI wiadomości jako całości

Wykład 3 – Szyfry strumieniowe, generatory oraz testy

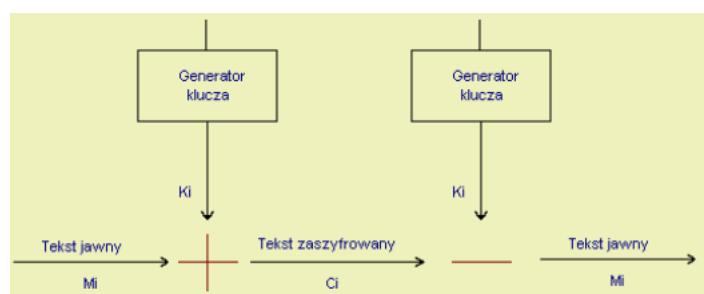
Szyfry strumieniowe:

- zazwyczaj są bardzo szybkie (szybsze niż szyfry blokowe) - używane tam gdzie szybkość jest ważna: WiFi, DVD, przesyłanie w czasie rzeczywistym mowy,
- PRNG jest, z definicji, nieprzewidywalny
 - mając n-bitów ciągu, nie jesteśmy przewidzieć n+1 bitu
- Słabe punkty:
 - nie zapewniają integralności
 - mają cechę łączności i przemienności ($X \oplus Y \oplus Z = (X \oplus Z) \oplus Y$,
 - $(P1 \oplus \text{PRNG}(K)) \oplus P2 = (P1 \oplus P2) \oplus \text{PRNG}(K)$
 - Bardzo groźny jest atak ze znanym tekstem jawnym, jeśli klucz (strumień klucza) się powtarza, jest okresowy
 - z własności XOR: $X \oplus X=0$,
 - $(P1 \oplus \text{PRNG}(IV,K)) \oplus (P2 \oplus \text{PRNG}(IV,K)) = P1 \oplus P2$
 - jeśli atakujący zna P1 łatwo jest wyznaczyć P2
 - nawet z $P1 \oplus P2$ można wyznaczyć tekst jawnego - przez redundancję języka - atak WPA2 Krack attack
- Bezpieczeństwo szyfrów strumieniowych:
 - zależy od zastosowanego generatora strumienia klucza szyfrującego
 - proces szyfrowania strumieniowego jest prostszy, niż w przypadku szyfrów blokowych - większa wydajność
 - w addytywnych szyfrach strumieniowych tekst zaszyfrowany powstaje jako wynik operacji sumowania modulo 2
 - redundancja związana z językiem naturalnym dla dobrze zaprojektowanych szyfrów strumieniowych jest bardzo mała, w przeciwieństwie do szyfrów blokowych
 - Nie gwarantują bezwzględnego bezpieczeństwa:
 - są tak bezpieczne jak użyty generator PRNG (PseudoRandom Number Generator)
 - jeśli używane/implementowane są prawidłowo, mogą być tak bezpieczne jak szyfry blokowe

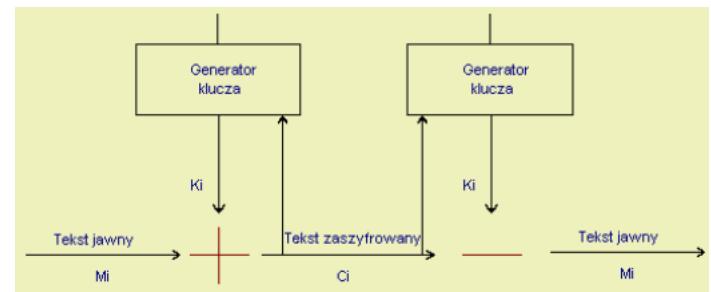


Rodzaje szyfrów strumieniowych:

- synchroniczne
 - strumień klucza w szyfrach strumieniowych synchronicznych generowany jest niezależnie od tekstu jawnego czy też szyfrogramu



- samosynchronizujące
 - każdy bit klucza jest zależny od pewnej stałej liczby n poprzednich bitów szyfrogramu
 - każdy znak tekstu jawnego wpływa na cały szyfrogram,
 - cechy statystyczne związane z tekstem jawnym są rozproszone w całym szyfrogramie (bardziej odporne na atak oparty na redundancji tekstu jawnego niż szyfry strumieniowe synchroniczne)
 - Wadą takiego trybu pracy jest propagacja błędów - modyfikacja jednego bitu podczas przesyłania spowoduje niepoprawne wytworzenie n bitów klucza po stronie deszyfrującej, a co jest z tym związane n błędnych bitów w tekście odszyfrowanym



Generatory kluczowe:

- Generatory kluczowe służą do generowania ciągów bitów, wykorzystywanych następnie jako klucze w strumieniowych algorytmach szyfrowania
- Mają one w większości przypadku wszystkie cechy statystyczne charakterystyczne dla ciągów losowych, lecz są powtarzalne, ze względu na to, iż są generowane przez algorytmy deterministyczne, korzystające jedynie z pewnych losowych danych wejściowych (ziarno losowe)
- Wielokrotne wywołanie tych algorytmów z takimi samymi danymi wejściowymi powoduje wygenerowanie takich samych wyjściowych ciągów. W przypadku kluczy kryptograficznych może to okazać się niebezpieczne, szczególnie, jeżeli kryptoanalytyk ma dostęp do generatora i zna klucz nadzędny!
- Przykładowymi elementami, które można wykorzystać do generowania liczb losowych są:
 - szum fal radiowych,
 - szum termiczny rezistorów i diod,
 - licznik Geigera-Müllera,
 - czas przybywania cząsteczek promieniowania kosmicznego,
 - temperatura rdzenia procesora.
- Przykład:
 - John von Neumann (1946) - metoda środka kwadratu
 - Zasada jej działania opiera się na podniesieniu do kwadratu poprzedniej liczby i wybraniu jej środkowej cyfry.
 - Analiza tej metody wykazała jednak, iż istnieje duże prawdopodobieństwo przejścia do ciągu o krótkim okresie
- Ogólna postać generatorów:
 - $X_{n+1} = aX_n^e + b \pmod{m}$, gdzie:
 - $n = 0, 1, \dots$
 - a, b, e, m są odpowiednio dobranymi liczbami naturalnymi
 - Dla $e=1$ oraz $b=0$:
 - $X_{n+1} = aX_n + b \pmod{m}$
 - otrzymujemy generator liniowy, dla którego m jest najczęściej potęgą liczby pierwszej lub podwojoną potegą liczby pierwszej, natomiast a równe jest pierwiastkowi pierwotnemu modulo m .

- Dzięki tak dobranym parametrom generator liniowy uzyskuje maksymalny okres.
- Dla $e=1$ oraz liczby b różnej od zera:
 - mamy do czynienia z generatorem afincznym.
 - Przy właściwym doborze parametrów a i b możemy uzyskać okres generatora równy m .
- Dla $e=2$ oraz $b=0$ otrzymujemy generator BBS:
 - W tym przypadku $m=pq$, gdzie p, q są różnymi liczbami pierwszymi dającymi w wyniku dzielenia przez 4 resztę 3.
- Dla $b=0$, biorąc dowolną liczbę e , ale przy ograniczeniu, że $\text{NWD}(e, m-1)=1$ i przy założeniu, że m jest iloczynem dwóch liczb pierwszych postaci $4k+3$ otrzymujemy generator RSA.

Liniowe generatory kongurencyjne:

- $X_n = (aX_{n-1} + b) \bmod m$, gdzie:
 - X_n – n -ty wyraz ciągu,
 - a – mnożnik (stała),
 - b – przyrost (stała),
 - m – moduł (stała),
 - X_0 – ziarno generatora.
- Cechy:
 - duża szybkość działania.
 - ciąg generowany przez ten generator będzie miał maksymalny okres równy m .
 - długość okresu uzależniony jest od doboru stałych a, b oraz m .
 - generatory są efektywne, a generowane ciągi cechują dobre właściwości statystyczne
 - cała rodzina generatorów kongruencyjnych (liniowe, kwadratowe, sześciennne, obcięte liniowe generatory) została złamana.

$$\varphi: N \rightarrow N$$

Funkcja Eulera:

- funkcja przypisująca każdej liczbie naturalnej liczبę liczb względnie pierwszych z nią i nie większych od niej

$$\varphi(1)=1$$

$$\varphi(2)=1$$

$$\varphi(3)=2$$

$$\varphi(4)=2$$

$$\varphi(5)=4$$

$$\varphi(6)=3$$

$$\varphi(7)=6$$

$$\varphi(8)=4$$

$$\varphi(n)=\text{card} \{ 0 \leq b < n \mid \text{NWD}(b, n) = 1 \}$$

1. p jest pierwsza a $\geq 1 \Rightarrow \varphi(p^a)=p^{a-1}(p-1)$
 - p jest pierwsza, a =1 $\Rightarrow \varphi(p) = p-1$
2. p, q względnie pierwsze $\Rightarrow \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q)$

$$\text{NWD}(k, n) = 1 \text{ i } kx \bmod n = 1 \Rightarrow x = k^{\varphi(n)-1} \bmod n$$

Generator BBS:

- Algorytm ten oparty jest na obliczaniu reszt kwadratowych modulo n
- 1. Bierzemy dwie duże liczby pierwsze p i q, które są kongruentne z 3 modulo 4 (muszą spełniać warunek $(p - 3) \bmod 4 = 0$ oraz $(q - 3) \bmod 4 = 0$). Iloczyn tych liczb n = p*q nazywamy liczbą Bluma.
 - Jeśli n jest liczbą Bluma, to każda reszta kwadratowa (mod n) ma dokładnie 4 pierwiastki kwadratowe.
- 2. W kolejnym kroku wybieramy inną losową liczbę całkowitą x, która jest względnie pierwsza z n ($1 \leq x \leq m-1$ i $\text{NWD}(m, x) = 1$).
- 3. Obliczamy wartość początkową dla generatora: $x_0 = x^2 \bmod n$
- 4. Elementem o numerze „i” w generowanym ciągu pseudolosowym jest najmniej znaczący bit x_i , gdzie x_i otrzymujemy z następującego wzoru: $x_i = x_{i-1}^2 \bmod n$.
 - Jeżeli chcemy obliczyć bit „i-ty” ciągu pseudolosowego bez wyznaczania wszystkich iteracji wcześniejszych, możemy skorzystać ze wzoru: $x_i = x_0^{2^i} \bmod ((p-1)(q-1)) \bmod n$

Generator RSA:

- $x_{i+1} = x_i^e \bmod N$, gdzie:
 - p, q – liczby pierwsze,
 - N – iloczyn liczb p i q,
 - e – liczba całkowita względnie pierwsza z $(p-1)(q-1)$,
 - x_0 – losowa wartość początkowa, $x_0 < N$
- Najmniej znaczący bit liczby x_i tworzy ciąg pseudolosowy
- Generator uważa się za bezpieczny, jeżeli liczba N jest dostatecznie duża

Algorytm potęgowania:

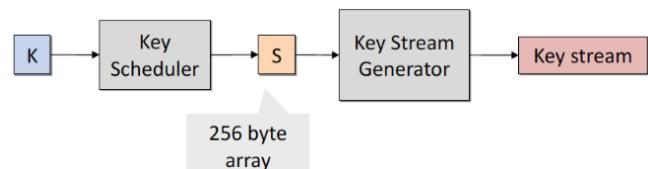
```
w:= 1
dla a = m do 1 // m - ilość miejsc binarnych liczby n
  c = a-ta cyfra binarna liczby n
  jeśli c = 0
    w:= w · w
  jeśli c = 1
    w:= w · w · x
```

RC4 Key Scheduler:

- Struktura szyfratora: 3 kroki:
 - Inicjalizacja:

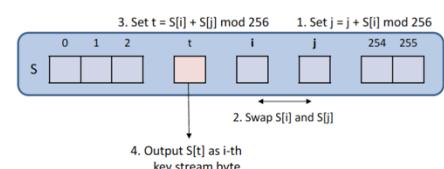

```
Char S[256];
int i;
for(i=0;i<256;i++)
  S[i] = i;
```
 - wyznaczanie klucza dla generatora


```
int i, j=0;
for(i=0;i<256;i++)
{
  j=( j + S[i] + T[i] ) mod 256;
  Swap(S[i], S[j]);
}
```
 - używany jest często wraz z wektorem IV (losowym lub jako wartość licznika), dołączanym do klucza



algorytm generowania ciągu losowego:

```
i=j=0;
while(true)
{
  i = ( i + 1 ) mod 256;
  j = ( j + S[i] ) mod 256;
  Swap( S[i], S[j] );
  t = ( S[i] + S[j] ) mod 256 ;
  k = S[t];
}
```



- RC4 jest niewystarczająco losowy! jego pierwszy bajt wygenerowanego ciągu zależy tylko od 3 komórek tablicy S, RSA sugeruje żeby pominąć pierwszych 256 bitów wygenerowanego ciągu
- Nie powinien być już używany!

Ciągi statystyczne losowe – testy statystyczne FIPS 140-2 (dla ciągów o długości 20 000 bitów):

- Test pojedynczych bitów:
 - $9725 < n(1) < 10275$
- Test serii (seria w danym ciągu to maksymalny podciąg następujących po sobie zer lub jedynek – **wybieramy czy szukamy zer czy jedynek, lub szukamy obydwoch serii i sprawdzamy dwukrotnie większe przedziały**):

Długość serii	Przedział
1	2315-2685
2	1114-1386
3	527-723
4	240-384
5	103-209
6 i więcej	103-209

- Test długiej serii:
 - serię zer albo jedynek nazywamy długą, jeśli ma długość 26 lub więcej
 - test zakończy się sukcesem jeśli w próbce o długości 20 000 bitów nie ma takiej serii
- Test pokerowy:
 - ciąg o długości 20 000 bitów należy podzielić na 5000 segmentów.
 - należy policzyć i zapamiętać liczbę wystąpień każdej z możliwych 16 - 4 bitowych wartości: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
 - Niech $s(i)$, gdzie $0 \leq i \leq 15$ oznacza liczbę wystąpień segmentów o wartości dziesiętnej i
 - Test zakończy się sukcesem, jeśli $2,16 < X < 46,17$, gdzie $X = \frac{16}{5000} * \sum_{i=0}^{15} s(i)^2 - 5000$

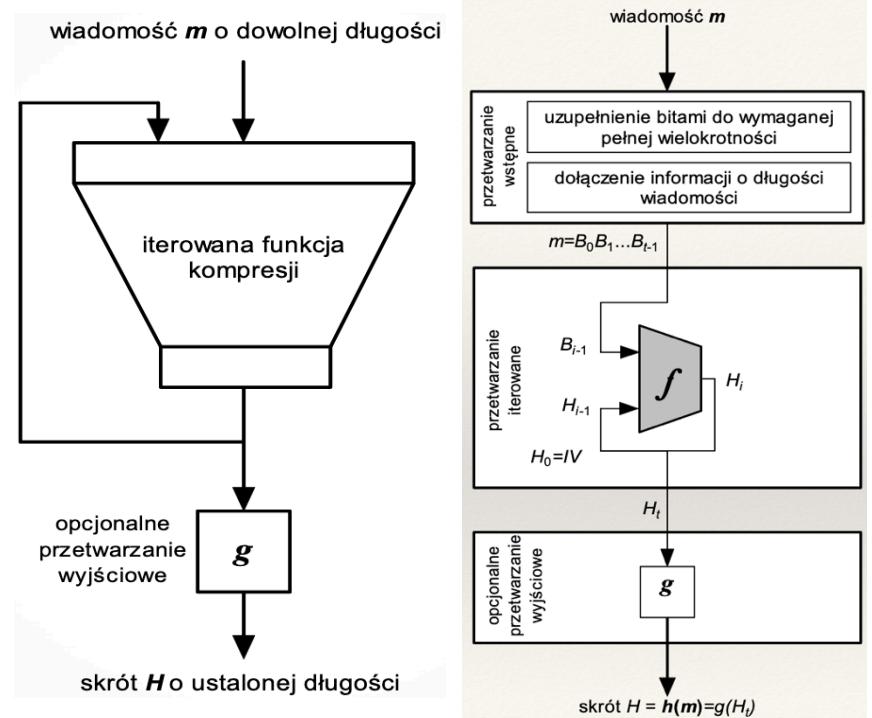
Podsumowanie:

- Pseudo Random Number Generators (PRNGs) używane są w kryptografii do: generowania klucza symetrycznego:
 - generowania klucza asymetrycznego lub parametrów w algorytmach generowania takiego klucza
- PRNGs używają generatorów pseudolosowych bitów - które generują po jednym losowym bicie
- Niektóre standardy używają pseudolosowych funkcji - Pseudo Random Function (PRF) zamiast PRNG
- PRNG przechodzi pozytywnie test następnego bitu jeśli nie istnieje wielomianowy algorytm, który pozwalałby na podstawie n-bitów wejściowych obliczyć kolejny bit $n+1$ z prawdopodobieństwem większym niż 0,5.

Wykład 4 – Funkcje skrótu:

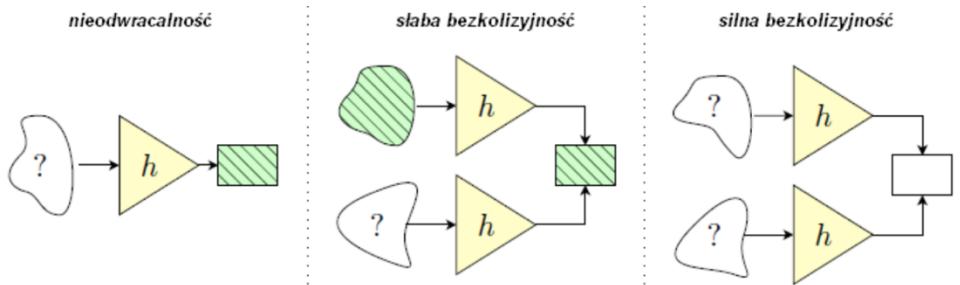
Jednokierunkowe funkcje skrótu:

- Funkcja $H(M) = h$, przy czym h jest liczbą o długości m .
- Argumentem jest wiadomość M o dowolnej długości



Własności funkcji skrótu:

- własność kompresji
- własność łatwości obliczeń:
 - mając dane M łatwo jest obliczyć h
- własność jednokierunkowości:
 - mając dane h , trudno jest obliczyć M (odporność na przeciwbraz) – nieodwracalność
- własność słabej odporności na kolizje:
 - mając dane M , trudno jest znaleźć inną wiadomość M' taką, że $H(M)=H(M')$ (słabo jednokierunkowa funkcja skrótu) (odporność na zastępczy przeciwbraz)
- własność odporności na kolizje:
 - jest obliczeniowo trudne znalezienie dwóch dowolnych argumentów $M \neq M'$, dla których $H(M)=H(M')$ (silnie jednokierunkowa funkcja skrótu) (odporność na kolizje)



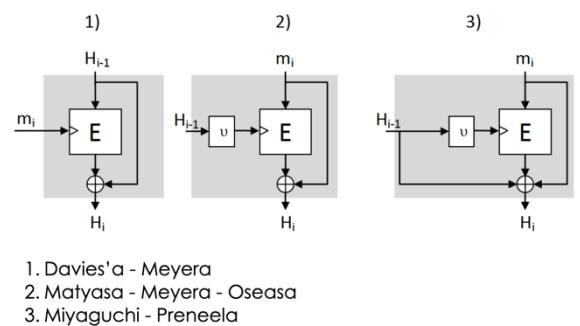
Dwie klasy funkcji skrótu:

- MDC - Manipulation Detection Code - funkcje skrótu bez klucza kryptograficznego
 - OWHF - odporne na przeciwbraz oraz zastępczy przeciwbraz
 - CRHF - odporne na kolizje (dodatkowo oprócz odporności na zastępczy przeciwbraz)
 - Trzy klasy ze względu na budowę:
 - z wykorzystaniem szyfrów blokowych
 - dedykowane - zaprojektowane wyłącznie do wyznaczania skrótu
 - w konstrukcji korzysta się z arytmetyki modularnej
- MAC – Message Authentication Code - funkcje skrótu z kluczem

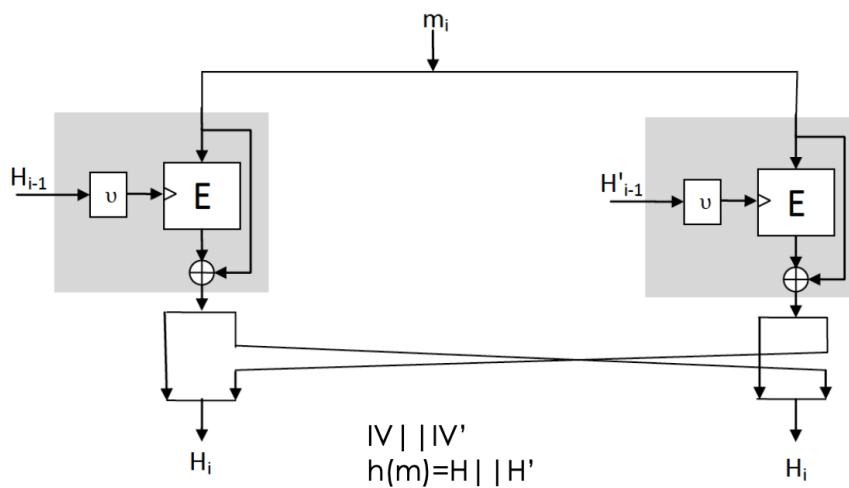
Funkcje skrótu z wykorzystaniem szyfrów blokowych:

- Używają trzech wstępnie ustalonych komponentów:
 - ogólny n-bitowy szyfr blokowy E którego parametrem jest klucz symetryczny k
 - funkcji v, która odwzorowuje n-bitowe wejścia w klucze k odpowiednie do użycia z E (jeśli klucze E mają także długości n, to g może być funkcją tożsamościową)
 - ustalona (zwykle n-bitowa) wartość początkowa IV, odpowiednia do użycia z E

Schematy funkcji skrótu wykorzystujące szyfry blokowe

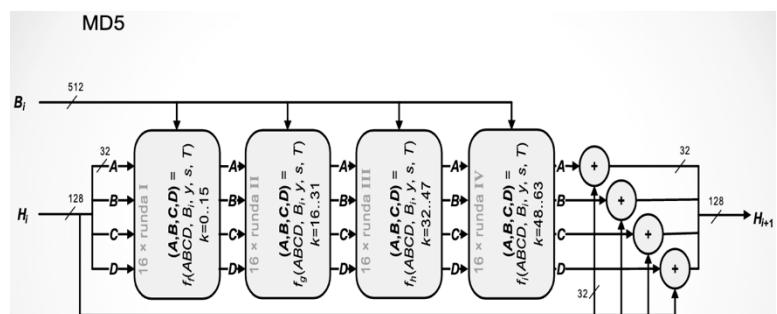


Schemat funkcji skrótu podwójnej długości MDC-2 (ISO/IEC 10118)



Funkcje dedykowane:

- Są to funkcje, które są specjalnie projektowane na potrzeby określonego celu skracania, z myślą o optymalizacji i szybkości
- przykłady: rodzina MD, SHA-1, RIPEMD- 128, 160.



Funkcje skrótu wykorzystujące arytmetykę modularną:

- Koncepcja:
 - takie zaprojektowanie iteracyjnej funkcji skrótu, aby arytmetyka mod M stanowiła podstawę funkcji kompresującej
- Uzasadnienie:
 - wykorzystanie istniejącego oprogramowania i sprzętu arytmetyki modularnej
 - skalowalność
- Przykład: MASH-1, MASH-2

MASH-1 (Modular Arithmetic Secure Hash)

$$H_i = ((x_i \oplus H_{i-1}) \vee A)^2 \pmod{N} \oplus H_{i-1}$$

$$A = 0xF00\dots00$$

x_i : first 4 bits in every byte equal to 1111 (1010 in last byte)
output transformation that reduces output size to at most $n/2$

MASH-2: replace exponent 2 by $2^8 + 1$

security for n -bit RSA modulus:

- best known attacks: preimage in $2^{n/2}$, collision in $2^{n/4}$
- feedforward of H_{i-1} essential

Funkcje skrótu z kluczem:

- Są podstawą dla dwóch rodzajów algorytmów kryptograficznych:
 - MAC - message authentication code - kod uwierzytelniania komunikatu - uwierzytelnia komunikat i chroni integralność
 - PRF - pseudorandom function - tworzą wyglądające na losowe wartości o długości skrótu

MAC:

- Tworzy wartość - tzw. znacznik uwierzytelniania komunikatu $M: T=MAC(K, M)$,
- gdzie jest stosowany:
 - dla każdego pakietu w:
 - IPsec
 - SSH
 - TLS
 - standardy telefonii kom. 3G i 4G szyfrują, ale nie uwierzytelniają
- Algorytmy kodu uwierzytelniania wiadomości:
 - MAC wykorzystujące szyfry blokowe - CBC-MAC
 - MAC zaprojektowane na podstawie algorytmów MDC (klucz dołączany jest jako część wejścia)

PRF:

- To funkcja, która używa tajnego klucza do wybrania i zwrócenia instancji: $PRF(K, M)$, gdy klucz jest tajny wynik jest nieprzewidywalny dla napastnika
- nie są używane samodzielnie, ale są częścią algorytmu kryptograficznego:
 - klucz w szyfratorze,
 - schematy identyfikacji wezwanie-odpowiedź,
 - 4G - uwierzytelnia kartę SIM i dostawcę usługi,
 - TLS - do wygenerowania klucza sesyjnego

Ataki na funkcję skrótu:

- Próba znalezienia M' , takiego że: $H(M)=H(M')$

Atak metodą urodzin:

- Paradoks urodzin:
 - jaka jest minimalna wartość k taka, że prawdopodobieństwo, iż co najmniej dwóch ludzi w grupie k ludzi ma urodziny tego samego dnia, wynosi więcej niż 0,5?
 - Prawdopodobieństwo, że inna osoba będzie miała tę samą datę urodzenia, w konkretnej grupie, jest niewielkie
 - Prawdopodobieństwo, że istnieje para osób w grupie (np. 23 osoby), która ma tę samą datę urodzenia jest większe niż 0,5
- W funkcji skrótu:
 - Oczekiwana liczba skrótów, jakie należy zbadać przed wystąpieniem koincydencji wynosi ok. $5,38 \cdot 10^9$ dla 64-bitowych skrótów
 - Jeśli skrót jest 128-bitowy, to próbując znaleźć dwie wiadomości o takim samym skrócie należy wykonać średnio $2,31 \cdot 10^{19}$ prób

- Strategia:
 - źródło zamierza podpisać komunikat przez dodanie doń odpowiedniej m-bitowej sumy kontrolnej i zaszyfrowanie jej kluczem prywatnym A
 - Przeciwnik generuje $2^m/2$ wersji komunikatu, z których wszystkie przekazują taką samą treść. Przygotowuje też taką liczbę komunikatów, z których wszystkie są wersjami komunikatu fałszywego, który ma zastąpić prawdziwy
 - Oba zbiory komunikatów zostają porównane w celu znalezienia pary, która da taki sam wynik działania funkcji haszującej. Prawdopodobieństwo sukcesu, według paradoksu dnia urodzin jest większe niż 0,5. Jeśli nie uda się znaleźć takiej pary, generuje się następne prawdziwe i fałszywe komunikaty aż do skutku
 - Przeciwnik przedstawia prawdziwą wersję A do sygnatury. Sygnaturę tę można dołączyć do wersji fałszywej i przesłać komunikat do adresata. Ponieważ obie wersje mają taką samą wartość funkcji haszującej, dadzątaką samą sygnaturę, przeciwnik może więc być pewien sukcesu mimo nieznajomości klucza do szyfru

Tripwire:

- stosuje funkcje skrótu do wykrywania zmian w plikach
- monitoruje prawa dostępu do plików, czasy modyfikacji plików oraz inne ważne właściwości plików (wybierane przez administratora)
- 4 tryby pracy:
 - tworzenia bazy danych
 - uaktualniania bazy danych
 - sprawdzania integralności
 - interaktywnego uaktualniania bazy danych

Zastosowania:

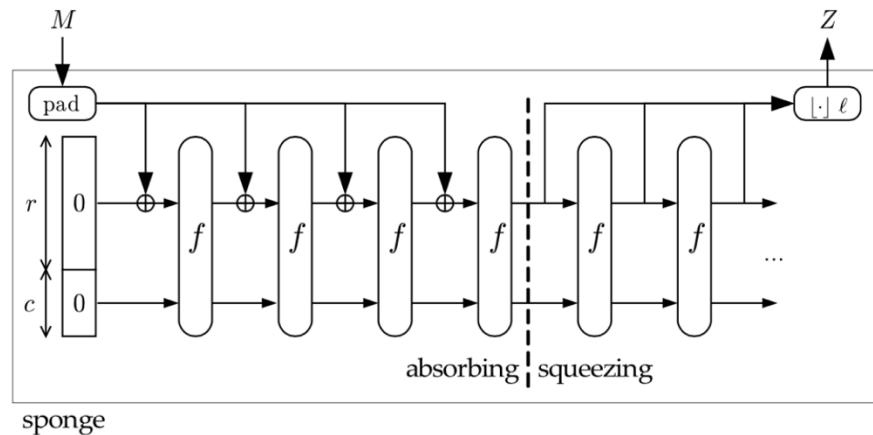
- Usługi certyfikacyjne
 - Protokoły certyfikacyjne. Wykorzystanie funkcji skrótu do schematów podpisu cyfrowego.
 - W schematach tych podpisywana jest nie sama wiadomość, ale właśnie jej skrót wyliczany za pomocą funkcji skrótu.
 - Innym wykorzystaniem są znaczniki czasowe wykorzystywane w usługach certyfikacyjnych
- Uwierzytelnienie
 - Grupa protokołów potwierdzająca tożsamość klienta.
 - Przykład: system Kerberos:
 - dostarcza mechanizmy uwierzytelniania, chroni przed podsłuchiwaniem oraz zapewnia integralność danych w modelu klient-serwer.
 - W systemie funkcje skrótu wykorzystywane są do wyliczania skrótu wprowadzanego przez klienta hasła, który to w dalszej części protokołu staje się tajnym kluczem klienta.
- Aktualizacje oprogramowania:
 - Systemy aktualizacji oprogramowania wykorzystują funkcje skrótu w celu weryfikacji integralności oprogramowania
- Bezpieczna komunikacja
 - Grupa zawierająca w sobie protokoły wymiany klucza.
 - Jednym z najpopularniejszych jest IPsec:

- Zapewnia on bezpieczną komunikację w warstwie sieciowej poprzez szyfrowanie i uwierzytelnianie pakietów IP.
 - Protokół wymiany klucza zastosowany w IPsec używa funkcji skrótu do generacji ciągów pseudolosowych.
 - Dodatkowo funkcja skrótu jest używana do weryfikacji integralności i uwierzytelnienia wszystkich wiadomości przesyłanych w tym protokole.
- Innym przykładem jest protokół SSH, który zapewnia uwierzytelnianie użytkowników i tworzy bezpieczny kanał pomiędzy lokalnym a zdalnym komputerem:
 - SSH wykorzystuje funkcje skrótu jako kody uwierzytelniające wiadomości.
 - To samo zastosowanie wykorzystane jest w protokole SSL/TLS.
- Bezpieczna poczta elektroniczna:
 - Kategoria zapewniająca bezpieczną wymianę wiadomości email, ochronę prywatności o sprawdzanie integralności wiadomości.
 - Protokół S/MIME używa funkcji skrótu w schemacie podpisu cyfrowego, zapewniając autentyczność i integralność przesyłanych wiadomości.
 - Również jedno z najpopularniejszych narzędzi do szyfrowania poczty elektronicznej – PGP, używa funkcji skrótu dla zapewnienia integralności wiadomości e-mail.
- Przechowywanie haseł:
 - Większość współczesnych systemów informatycznych nie przechowuje haseł w postaci jawnej, lecz wykorzystuje funkcje skrótu do tworzenia skrótów haseł, które następnie są przechowywane w systemie.
 - Rozwiązania takie spotykane są powszechnie w większości systemów baz danych oraz w systemach operacyjnych.

KECCAK:

- Zwycięzca konkursu na SHA-3 z 2012 roku
- Funkcja rundy:

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$
- Ilość rund: 12+21
- Wydajność:
 - Wysoki poziom zrównoleglenia
 - Elastyczność – przeplatanie bitów
 - Szybszy niż SHA-2 na każdym współczesnym komputerze
- Budowa:
 - Keccak ma architekturę „gąbki” (ang. sponge construction) — bloki wejściowe są stopniowo „wchłaniane” w kolejnych etapach i mieszane z dużym rejestrem stanu.
 - Blok wyjściowy jest konstruowany w podobny sposób, przez „wyciskanie” kolejnych fragmentów danych wyjściowych z rejestrów stanu, wielokrotnie go mieszając pomiędzy wyciskanymi blokami.
 - Wchłanianie i wyciskanie odbywa się na małej części rejestrów stanu poprzez wykonanie funkcji binarnej alternatywy wykluczającej (xor) z danymi wejściowymi lub odczyt tej samej małej części przy odczytce danych wyjściowych.
 - Pozostała część stanu nigdy nie jest bezpośrednio używana do konstrukcji danych wyjściowych ani nie oddziaływa bezpośrednio z danymi wejściowymi.



- Funkcja mieszająca:
 - Funkcja mieszająca stanu (funkcja f na rysunku), składa się z wielokrotnej aplikacji funkcji rundy (do 24 razy w przypadku największej wersji algorytmu).
 - Każda runda z kolei składa się z kompozycji 7 prostych i wydajnych w implementacji funkcji, które dokonują odwracalnych permutacji, rotacji, mieszań albo dyfuzji.
 - Ostatnia z tych funkcji w rundzie dodatkowo jest parametryzowana stałą wartością zależną od numeru rundy (i wersji algorytmu) w celu usunięcia symetrii z funkcji rundy.
- Dane wejściowe są dopełniane do całkowitej wielokrotności liczby bitów w pojedynczym bloku przy pomocy prostego schematu dopełniania zwanego „101”:
 - do ciągu danych wejściowych w postaci ciągu binarnego dopisz bit o wartości 1, następnie bity o wartości 0, aż do przedostatniego bitu w pełnym bloku, następnie dopisz bit o wartości 1 dopełniając blok całkowicie.
- Porównanie bezpieczeństwa-szybkość:
 - Instancja o $r = 1088$ i $c=512$:
 - Szerokość permutacji: 1600
 - Siła bezpieczeństwa 256 -> wystarczająca „post-quantum”
 - Instancja o $r=40$ i $c=160$:
 - Szerokość permutacji: 200
 - Siła bezpieczeństwa 80 -> taka sama jak w SHA-1

Jak zbudować silną permutację:

- Zbudować jak iteracyjną permutację
- Jak szyfr blokowy:
 - Sekwencja identycznych rund
 - Runda składa się z sekwencji prostych mappingów
- ALE:
 - Bez kolejności kluczy
 - Stałe rundy zamiast klucznych rundy
 - Odwrotna permutacja może nie wystarczyć
- Klasyczne kryteria LC/DC:
 - brak dużych różnicowych prawdopodobieństw propagacji
 - brak dużych korelacji między wejściem i wyjściem
- Niewykonalność problemu CICO (Constrained Input Constrained Output):
 - Mając częściowe wejście i wyjście nie da się znaleźć brakujących części
- Odporność na:
 - Integralną kryptoanalizę
 - Ataki algebraiczne
 - Ataki typu slide i symmetry

Aplikacje stosujące KECCAK:

- Regularne haszowanie:
 - Elektroniczne podpisy
 - Integralność danych
 - Indentyfikator danych
- Hasz solony:
 - Losowe generowanie funkcji skrótu

- Składowanie haseł i weryfikacja
- Funkcja generowania maski:
 - Funkcja wyprowadzania klucza w SSL, TLS
 - haszowanie pełnej domeny w kryptografii klucza publicznego
- Kody uwierzytelniające wiadomości:
 - Kod autentykacji wiadomości
- Szyfrowanie strumieniowe:
 - Jako szyfr strumieniowy
- Szyfrowanie z uwierzytelnianiem jednoprzebiegowym:
 - Autentykacja i szyfrowanie w pojedynczym przebiegu
 - Bezpieczne wysyłanie wiadomości (SSH, SSL/TLS, IPSEC)

Wykład 5 – Kryptografia asymetryczna

Testy pierwszości:

- deterministyczne
- probabilistyczne

Test Fermata:

- Wyznacz a takie, że $1 \leq a < p$
- Sprawdź k losowych wartości a
- Jeśli $(a^{p-1}-1) \bmod p = 0$, dla każdego $k \rightarrow$ liczba p jest pierwsza
- Złożoność: $O(k \log^3 n)$

Liczby Carmichaela:

- Przechodzą wszystkie testy Fermata
- Nieparzyste
- Przy faktoryzacji maksymalna potęga czynnika to 1
- Iloczyn minimum 3 liczb pierwszych
- 1 taka liczba na 160 bilionów liczb naturalnych
- Przykład: $561 = 3 * 11 * 17$

Test Solovaya-Strassena (probabilistyczny):

- K iteracji:
 - Wybierz a ze zbioru $\{1, \dots, n-1\}$
 - $X = a/n \rightarrow$ Symbol Jacobiego
 - Jeśli $x=0$ lub $a^{(n-1)/2} \pmod{n} \neq x$:
 - Zwróć: **złożona** (a – świadek Eulera)
- Zwróć: **prawdopodobnie pierwsza**
- Dla dowolnej liczby nieparzystej n , co najmniej $\frac{1}{2}$ wartości a świadczą o jej złożoności
- Złożoność czasowa: $O(k \log^3 n)$
- Prawdopodobieństwo false-positive: 2^{-k}

Test Millera-Rabina (probabilistyczny):

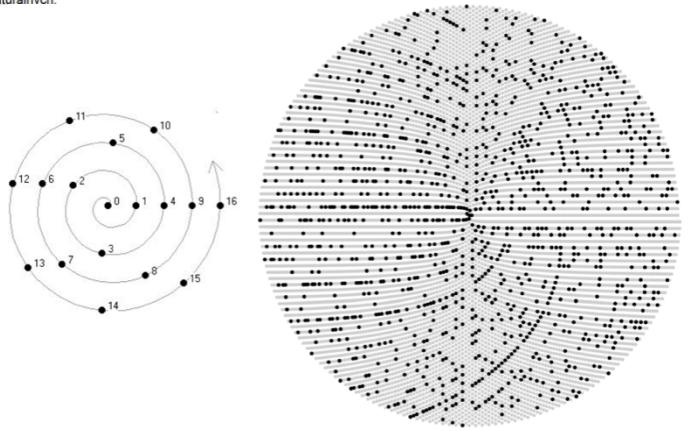
- Wyznacz s – maksymalną potęgę dwójki dzielącą $n-1$
- $d = (n-1)/2^s$
- k iteracji:
 - wybierz a ze zbioru $\{1, \dots, n-1\}$
 - jeśli:
 - $a^d \pmod{n} \neq 1$ oraz $a^{2^r d} \pmod{n} \neq n-1$ dla r należącego do $\{0, \dots, s-1\}$:
 - zwróć: **złożona**
- Zwróć: **prawdopodobnie pierwsza**
- Dla dowolnej liczby nieparzystej n , co najmniej $\frac{3}{4}$ wartości a świadczą o jej złożoności
- Złożoność czasowa: $O(k \log^4 n)$
- Prawdopodobieństwo false positive: 4^{-k}

Sito Atkina-Bernsteina:

- Wszystkie liczby dające resztę z dzielenia przez 60 równą 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56 lub 58 są podzielne przez 2, zatem nie są pierwsze – algorytm je ignoruje.
- Wszystkie liczby dające resztę z dzielenia przez 60 równą 3, 9, 15, 21, 27, 33, 39, 45, 51 lub 57 są z kolei podzielne przez 3 i również nie są pierwsze – algorytm je ignoruje.
- Wszystkie liczby dające resztę z dzielenia przez 60 równą 5, 25, 35 lub 55 są podzielne przez 5 i także nie są pierwsze – algorytm je ignoruje.
- Wszystkie liczby dające resztę z dzielenia przez 60 równą 1, 13, 17, 29, 37, 41, 49 lub 53 posiadają resztę z dzielenia przez 12 równą 1 lub 5. Liczby te są pierwsze wtedy i tylko wtedy, gdy liczba rozwiązań równania $4x^2 + y^2 = n$ jest nieparzysta dla $x, y \in \mathbb{N}$, a liczba n jest niepodzielna przez kwadraty liczb naturalnych.
- Wszystkie liczby dające resztę z dzielenia przez 60 równą 7, 19, 31 lub 43 posiadają resztę z dzielenia przez 12 równą 7. Są one liczbami pierwszymi wtedy i tylko wtedy, gdy liczba rozwiązań równania $3x^2 + y^2 = n$ jest nieparzysta dla $x, y \in \mathbb{N}$, a liczba n jest niepodzielna przez kwadraty liczb naturalnych.
- Wszystkie liczby dające resztę z dzielenia przez 60 równą 11, 23, 47 lub 59 posiadają resztę z dzielenia przez 12 równą 11. Są one liczbami pierwszymi wtedy i tylko wtedy, gdy liczba rozwiązań równania $3x^2 - y^2 = n$ jest nieparzysta dla $x, y \in \mathbb{N}$, a liczba n jest niepodzielna przez kwadraty liczb naturalnych.

Spirala Ulama:

- graficzna metoda pokazywania pewnych niewyjaśnionych do dziś prawidłowości w rozkładzie liczb pierwszych
- Na kwadratowej tablicy zaczynając od 1 w środku spiralnie wypisuje się kolejne liczby naturalne. Na niektórych przekątnych liczby pierwsze grupują się częściej niż na innych. Fakt ten nie został do tej pory wyjaśniony. Zjawisko występuje także, jeśli rozpoczyna się od innych wartości niż 1.
- Spirala Sacksa – wariant spirali Ulama



Liczby Mersenne'a:

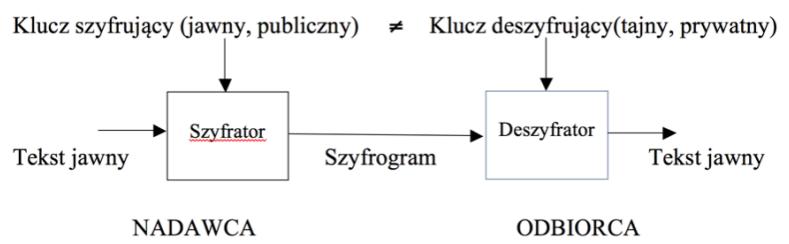
- Jeśli n jest liczbą pierwszą, to $M_n = 2^n - 1$ również może nią być.

Postulat Bertrandta:

- W przedziale $(1..n)$ jest $\pi(n) \approx n/\ln(n)$ liczb pierwszych
- Pomiędzy liczbą p_n a $2*p_n$ jest przynajmniej jedna liczba pierwsza

Algorytmy asymetryczne:

- W najprostszym ujęciu problem składa się z:
 - Zbioru wiadomości M
 - Zbioru kryptogramów C
 - Algorytmu (probabilistycznego) G generacji klucza, który generuje parę (e, d)
 - Algorytmu szyfrującego E (deterministycznego), który na wejściu e i $m \in M$ zwraca kryptogram $E(e, m)$
 - Algorytmu odszyfrowującego D , który na wejściu d i c zwraca $D(d, c) \in M$
- Oczywiście: $D(d, E(e, m)) = m$
- Schemat podpisu składa się z:
 - Zbioru wiadomości W
 - Zbioru podpisów T
 - Algorytmu (probabilistycznego) generacji klucza, który generuje parę (e, d)
 - Algorytmu podpisującego S , który na wejściu d i $m \in W$ zwraca podpis elektroniczny $S(d, m)$
 - Algorytmu weryfikacji V , który na wejściu d, s, m zwraca tak albo nie (w przypadku, gdy $s = S(e, m)$, algorytm powinien zwrócić tak)



Algorytm Diffiego-Hellmana:

- Wykorzystywany do dystrybucji kluczy
 - 1. Uzgodnij w sposób jawnym wybór dwóch dużych liczb całkowitych n i g : $1 < g < n$
 - Warunki bezpieczeństwa:
 - $n \in (n-1)/2 \rightarrow$ liczby pierwsze o długości 512 (lub nawet 1024) bitów
 - g – pierwiastek pierwotny modulo n
 - 2. Osoba A wybiera losowo dużą liczbę całkowitą x (tajną) i oblicza: $X = g^x \bmod n$
 - 3. Osoba B wybiera losowo dużą liczbę całkowitą y (tajną) i oblicza: $Y = g^y \bmod n$
 - 4. Osoba A wysyła X do B, a osoba B wysyła Y do A
 - 5. Osoba A oblicza $k = Y^x \bmod n$
 - 6. Osoba B oblicza $k' = X^y \bmod n$
 - 7. $K = g^{xy} \bmod n = k' = k$
- Alice and Bob share the key $K = 4$
- Rozszerzony algorytm Diffiego-Hellmana:
 - Jest jeszcze Z , a X, Y, Z jest przesyłane w kótku, każdy oblicza kolejno:
 - $Z' = Z^x \bmod n$
 - $X' = X^y \bmod n$
 - $Y' = Y^z \bmod n$
 - Następnie jeszcze raz przesyłane w kótku i każdy oblicza:
 - $k = Y'^x \bmod n$
 - $k = Z'^y \bmod n$
 - $k = X'^z \bmod n$
 - $k = g^{xyz}$

RSA:

- pierwszy opublikowany algorytm z kluczem publicznym
 - Bezpieczeństwo opiera się na trudności faktoryzacji dużych liczb
 - Generowanie kluczy:
 - Wybieramy dwie duże liczby pierwsze p i q i obliczamy iloczyn:
 - $N = p * q$ oraz $\phi = (p-1) * (q-1)$
 - Losowo wybieramy klucz szyfrujący e tak, że e i ϕ są względnie pierwsze
 - Wyznaczamy klucz deszyfrujący d za pomocą algorytmu Euklidesa z zależności:
 - $(E * d - 1) \bmod \phi == 0$
 - Szyfrowanie:
 - Tekst jawny m dzielimy na bloki liczbowe m_i mniejsze od n
 - Szyfrogram obliczamy ze wzoru:
 - $C_i = m_i^e \bmod n$
 - Deszyfrowanie:
 - $M_i = C_i^d \bmod n$,
 - Ponieważ: $C_i^d = (m_i^e)^d = m_i^{ed} = m_i^{-1} = m_i \bmod n$
 - Bezpieczne długości n : 512 bitów, 664 bity, 1024 bity
- Generowanie kluczy:**

Wybieramy:	$p = 47$ i $q = 71$
Obliczamy:	$n = 47 \times 71 = 3337$
	$\phi = 46 \times 70 = 3220$

Wybieramy (losowo): $e = 79$ (względnie pierwsze z ϕ)
 Obliczamy (Euklides): $d = 79^{-1} \bmod 3220 = 1019$
 Publikujemy e i n , utajniamy d i usuwamy p i q .
- Szyfrowanie:**
 Wiadomość $m = 6882326879666683$ dzielimy na bloki:
 $m_1 = 688, m_2 = 232, m_3 = 687, m_4 = 966, m_5 = 668, m_6 = 3.$
 $c_1 = 688^{79} \bmod 3337 = 1570$ stąd
 $c = 1570\ 2756\ 2091\ 2276\ 2423\ 158$
- Deszyfrowanie:**
 $m_1 = 1570^{1019} \bmod 3337 = 688$

- Niebezpieczeństwa: wykorzystywanie jednego n dla kilku, małe wykładniki e lub d ($d < 1/4n$), generowanie liczb pierwszych p i q (testy probabilistyczne)

Algorytm ElGamala:

- Oparty na trudności obliczania logarytmów dyskretnych
- Generowanie kluczy:
 - Wybieramy liczbę pierwszą p i dwie liczby losowe:
 - g (pierwiastek pierwotny mod p)
 - x ($x < p-1$)
 - obliczamy $y = g^x \text{ mod } p$
 - klucz jawny y, g i p; klucz prywatny: x
- Szyfrowanie:
 - Wybieramy losową liczbę k $< p-1$
 - Obliczamy:
 - $a = g^k \text{ mod } p$
 - $b = y^k M \text{ mod } p$
 - Wysyłamy jako szyfrogram parę liczb a i b, $c = \{a, b\}$ (szyfrogram jest dwukrotnie dłuższy od wiadomości)
- Deszyfrowanie:
 - Obliczamy:
 - $M = \frac{b}{a^x} \text{ mod } p$, bo: $a^x = g^{kx} \text{ mod } p$ i
 $\frac{b}{a^x} \equiv \frac{y^k M}{a^x} \equiv \frac{g^{kx} M}{g^{kx}} \equiv M \text{ mod } p$

Generowanie kluczy:

Wybieramy:	$p = 2357, g = 2, x = 1751$
Obliczamy:	$y = g^x \text{ mod } p = 2^{1751} \text{ mod } 2357 = 1185$
Klucz publiczny:	$p = 2357, g = 2, y = 1185,$
Klucz prywatny:	$x = 1751$

Szyfrowanie:

wiadomość:	$m = 2035$
wybieramy:	$k = 1520$
obliczamy:	$a = 2^{1520} \text{ mod } 2357 = 1430$
	$b = 2035 \cdot 1185^{1520} \text{ mod } 2357 = 697$
wysyłamy:	$c = \{1430, 697\}$

Deszyfrowanie:

$$1/a^x = a^{-x} = a^{p-1-x} = 1430^{605} \text{ mod } 2357 = 871$$

$$m = 871 \cdot b \text{ mod } p = 871 \cdot 697 \text{ mod } 2357 = 2035$$

Szyfrowanie Rabina:

- pierwszy przykład udowodnionego bezpieczeństwa
- odtworzenie tekstu jawnego na podstawie określonego szyfrogramu jest równoważne z rozkładem na czynniki
- Oparty na trudności znalezienia pierwiastków kwadratowych modulo liczba złożona, równoważny problemowi faktoryzacji
- Generowanie kluczy:
 - klucz tajny: liczby pierwsze p i q kongruentne do 3 modulo 4
 - takie, że $(p - 3) \text{ mod } 4 == 0$ oraz $(q - 3) \text{ mod } 4 == 0$
 - klucz jawny: $n = pq$
- Szyfrowanie:
 - Wiadomość $M < n$: $C = M^2 \text{ mod } n$
- Deszyfrowanie:
 - $M_1 = C^{(p+1)/4} \text{ mod } n$
 - $M_2 = p - C^{(p+1)/4} \text{ mod } n$
 - $M_3 = C^{(q+1)/4} \text{ mod } n$
 - $M_4 = q - C^{(q+1)/4} \text{ mod } n$

Przykład:

Klucze: prywatny $p = 277, q = 331$, publiczny: $n = pq = 91687$;

Wiadomość: $m = 1001111001 111001$ $m = 40569$;

Szyfrowanie:

$$c = m^2 \text{ mod } n = 40569^2 \text{ mod } 91687 = 62111$$

Deszyfrowanie:

$$m_1 = 69654, m_2 = 22033, m_3 = 40569, m_4 = 51118,$$

$$m_1 = 1000100000 010110, m_2 = 101011000 010001,$$

$$\underline{m_3 = 1001111001 111001}, m_4 = 1100011110 101110.$$

Algorytm plecakowy:

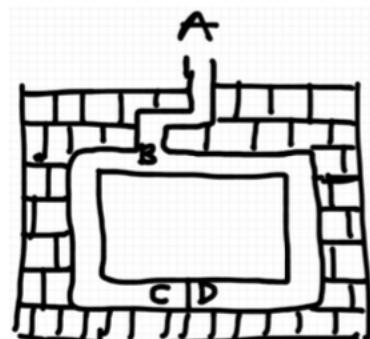
- Oparty na problemie plecaka NP-zupełnym.
- za pomocą mnożenia modularnego i permutacji algorytm próbuje zamaskować łatwo rozwiązywalny problem sumy podzbiorów, nazywany problemem sumy podzbioru superrosnącego
- Tworzenie kluczy:
 - klucz prywatny: wybieramy ciąg dla plecaka superroszącego np.: 2, 3, 6, 13, 27, 52;
 - wybieramy moduł m większy od sumy wszystkich liczb w ciągu (np. m = 105) i mnożnik n nie mający wspólnych dzielników z żadną z liczb w ciągu (np. n = 31);
 - klucz jawny: mnożymy wszystkie liczby ciągu klucza prywatnego razy n mod m ($2 \times 31 \text{ mod } 105 = 62, 93, 81, 88, 102, 37$)
- Szyfrowanie: Tekst jawny: 011000 110101 101110
Szyfrowanie: $93 + 81, 62 + 93 + 88 + 37, 62 + 81 + 88 + 102$
Szyfrogram: 174, 280, 333
- Deszyfrowanie: Określić n^{-1} takie, że $n \times n^{-1} = 1 \text{ mod } m$ ($n^{-1} = 61$).
Szyfrogram: 174, 280, 333
Przekształcenie: $174 \times 61 \text{ mod } 105 = 9$
9 70 48
Deszyfrowanie: $3 + 6, 2 + 3 + 13 + 52, 2 + 6 + 13 + 27$
Tekst jawny: 011000 110101 101110

Zastosowania algorytmów klucza publicznego:

- podpisy cyfrowe;
- protokoły identyfikacji;
- protokoły uwierzytelniania (autentyczności); protokoły wymiany klucza (Diffie–Hellman); dowody o zerowej wiedzy;
- i inne.

Podstawowy protokół o wiedzy zerowej:

- Peggy zna tajemnicę jaskini. Pragnie wykazać swoją wiedzę Victorowi, ale nie chce ujawnić zaklęcia.
1. Victor stoi w punkcie A.
 2. Peggy przechodzi całą drogę w jaskini do jednego z punktów C lub D
 3. Kiedy Peggy znika w jaskini, Victor przechodzi do punktu B
 4. Victor krzyczy do Peggy, żądając aby:
 - a. wyszła z lewego korytarza; albo
 - b. wyszła z prawego korytarza



5. Peggy stosuje się do polecenia, używając zaklęcia do otwarcia drzwi tajemnych
6. Peggy i Victor n-krotnie powtarzają kroki 1-5

Czy Victor może przekonać trzecią osobę o poprawności dowodu? Jaka szansę ma Peggy na oszukanie Victora?

Dowód o wiedzy zerowej w przypadku RSA:

- Alice zna klucz prywatny Carol, chce przekonać Boba że ma ten klucz, bez zdradzania go.
- Kluczem publicznym Carol jest e, prywatnym d, modułem RSA n.
- Alice i Bob uzgadniają liczby losowe k i m takie, że $km \equiv e \pmod{n}$.
 - Muszą wybrać te liczby losowo, najpierw k, a potem dopasować m. Obie liczby muszą być > 3 .
- Alice i Bob tworzą szyfrogram losowy C
- Alice stosując klucz prywatny Carol oblicza $M = C^d \pmod{n}$, następnie oblicza $X = M^k \pmod{n}$ i przesyła X Bobowi
- Bob potwierdza, że $X^m \pmod{n} = C$.
- Jeśli wynik się zgadza to Bob wierzy Alice.

Protokół związany z podpisami (ślepe podpisy):

- Alicja zakrywa wiadomość m: w tym celu wybiera losową liczbę k $< n$ względnie pierwszą z n i oblicza $t = m * k^e \pmod{n}$.
- Alicja przesyła liczbę t notariuszowi
- Notariusz szyfruje t za pomocą swego prywatnego klucza: $s = td \pmod{n}$.
- Notariusz przesyła Alicji liczbę s.
- Ponieważ $s = t^d = (m * k^e)^d = m^d * k^{ed} = m^d * k \pmod{n}$:
 - Alicja może łatwo obliczyć $m^d = s/k \pmod{n}$.
 - Ale $m^d \pmod{n}$ jest podpisana wiadomością m.

Obliczenia wielostronne:

- Problem jest trywialny jeśli istnieje zaufana strona trzecia
- Algorytmem wykorzystywanym przy bezpiecznych obliczeniach wielostronnych może być RSA

Problem milionerów:

- Dwoje milionerów, nazwijmy ich Apolonią i Bogusiem, pragnie ustalić, kto z nich jest bogatszy. Żadne z nich nie zamierza jednak zdradać drugiej stronie – ani nikomu innemu – wartości swojego majątku
- Apolonia ma 1 milionów, Boguś ma J milionów. Poufnymi danymi wejściowymi są więc w protokole I i J, obliczaną funkcją – funkcja określająca, czy $I > J$.
- Przyjmujemy, że są to liczby z określonego przedziału: $I, J \in [1..M]$ (przyjmując odpowiednio duże M nie zdradzimy dodatkowych informacji o I i J)

Protokół rozwiążający problem milionerów przebiega następująco:

Apolonia	Boguś
$I \in [1..M]$	$J \in [1..M]$
1) $n, e, d = e^{-1} \bmod \phi(n)$ – kl. prywatny $n, e, d = e^{-1} \bmod \phi(n) - \text{kl. prywatny}$	$e, n - \text{kl. publiczny}$
2)	\xleftarrow{m} losowy x $C = E_{e,n}(x) = x^e \bmod n$ $m = C - J + 1$
3) $Y_i = D_{d,n}(m + i - 1)$ $= (m + i - 1)^d \bmod n, i \in [1..M]$	
4) losowe $p \ll n$ $Z_i = Y_i \bmod p, i \in [1..M]$	
5) $W_j = Z_i + (i \geq I) \bmod p, i \in [1..M]$	$\xrightarrow{p, W_1, \dots, W_M}$ wynik : $W_J \equiv x \bmod p$ $\xleftarrow{\text{wynik}}$ (nie $\Leftrightarrow I < J$)
6)	

1. Apolonia generuje klucze RSA; klucz publiczny (n, e) który udostępnia Bogusowi i swój tajny klucz prywatny d .
2. Boguś wybiera losową wartość x . Szyfruje x za pomocą klucza publicznego Apolonii, czyli oblicza $C = x^e \bmod n$, następnie wysyła do Apolonii wartość $m = C - J + 1$. Zauważmy, że dla Apolonii, która nie zna x , otrzymana liczba nie powie nic o tajnej wartości J Bogusia.
3. Apolonia przeprowadza teraz – dla wszystkich możliwych wartości J Bogusia – obliczenia odwrotne. Dla liczb $j \in [1..M]$ oblicza wartość $Y_j = (m + j - 1)^d \bmod n$. W przypadku gdy $j = J$ Apolonia otrzyma w wyniku x , w wszystkich pozostałych – przypadkowe wartości.
4. Apolonia oblicza $Z_j = Y_j \bmod p$ dla wszystkich $j \in [1..M]$. Należy p dobrać w taki sposób, aby wszystkie zredukowane wartości różniły się co najmniej o 2. Redukcja, która jest operacją jednokierunkową, uniemożliwi później Bogusowi przeprowadzenie obliczeń odwrotnych.
5. Apolonia dodaje 1 do wszystkich Z_j o indeksie większym lub równym I . Następnie przekazuje Bogusowi ciąg wartości W_1, W_2, \dots, W_M , czyli $Z_1, Z_2, \dots, Z_{I-1}, Z_I + 1, Z_{I+1} + 1, \dots, Z_M + 1$ i liczbę p . Zauważmy, że Boguś nie jest w stanie odtworzyć tajnego I Apolonii, czyli miejsca w którym W_j przestaje być równe Z_j . Aby to zrobić, Boguś musiałby odtworzyć obliczenia Apolonii, czyli najpierw złamać RSA.
6. Boguś sprawdza, czy $W_J \equiv x \bmod p$. Jeżeli nie, to znaczy że Apolonia zaczęła dodawać jedynkę do wartości Z_j dla indeksów mniejszych niż J , czyli $I < J$. Tą wiadomością Boguś dzieli się z Apolonią.

Problem miłosny:

- Apolonia kocha Bogusia. Jeżeli Apolonia wyzna Bogusowi miłość, a uczucie jest wzajemne, wtedy wszystko skończy się happy endem. Jeżeli natomiast Boguś nie kocha Apolonii, to Apolonia będzie niezadowolona, że niepotrzebnie zdobyła się na to wyznanie. Czy możemy pomóc Apolonii? Innymi słowy, czy możemy skonstruować protokół, dzięki któremu Apolonia sprawdzi, czy Boguś ją kocha, nie wyznając mu niepotrzebnie swoich uczuć. Chcemy też, żeby protokół był symetryczny, tzn. pozwolił na to samo Bogusowi.
- Równoważnym problemem jest bezpieczne obliczanie iloczynu logicznego. Każda ze stron A i B ma swój sekretny bit – a i b. Jeżeli obliczymy w sposób bezpieczny wartość $a \wedge b$ i ujawnimy obu stronom wynik, to A pozna b tylko wtedy, gdy $a = 1$; analogicznie dla B. Przekładając to na język pierwszego problemu – niech 0 oznacza „nie kocha”, a 1 – „kocha”. Boguś poznając wartość $a \wedge b$ owie się, że Apolonia go kocha tylko wówczas, gdy on sam odwzajemnia jej uczucie.
- Przekaz nierozróżnialny „1 z 2”. Skonstruujmy najpierw protokół, który działa następująco:

A	B
sekrety s_0, s_1	tajny bit $b \in \{0, 1\}$ – (bit wyboru)
$\xrightarrow{s_b}$	$b = ?$

- Pozwala on na przekazanie dokładnie jednego, wybranego przez B sekretu strony A stronie B, przy czym A nie wie się, który sekret został przekazany. Jest to tzw. przekaz nieroróżnialny 1 z 2 (1 out of 2 oblivious transfer)

	Apolonia	Boguś
	sekrety s_0, s_1	bit $b \in \{0, 1\}$
1)	Klucz RSA: n, e, d losowe x_0, x_1	
2)		$n, e, x_0, x_1 \xrightarrow{q} \text{losowe } k$
3)	$t_0 = s_0 + ((q - x_0)^d \bmod n)$ $t_1 = s_1 + ((q - x_1)^d \bmod n)$	$q = (k^e \bmod n) + x_b$
4)		$\xleftarrow{q} t_0, t_1 \xrightarrow{t_b} t_b - k = s_b$

Apolonia ma dwa sekrety (zakodowane do postaci liczb): s_0 i s_1 . Boguś ma swój tajny bit b , odpowiadający numerowi sekretu Apolonii, który chce poznać. Protokół przebiega następująco:

- Apolonia generuje klucz publiczny i prywatny RSA oraz dwie losowe liczby x_0 i x_1 . Klucz publiczny, czyli parę (n, e) oraz wartości x_0 i x_1 przekazuje Bogusowi.
- Boguś szyfruje losowo wybraną liczbę k za pomocą klucza publicznego Apolonii i dodaje do wyniku tej z losowych liczb Apolonii, która odpowiada wartości bitu b – np. oblicza $k^e x_0$, jeśli $b = 0$. Boguś przesyła tę wartość Apolonii. Odnotujmy, że nie da jej to możliwości poznania b , gdyż nie zna ona wartości k wykorzystanej przez Bogusia w obliczeniach.
- Apolonia wykonuje teraz obliczenia dla obu możliwych wartości b . Oblicza $t_i = s_i + ((q - x_i)^d \bmod N)$ dla $i = 0$ i dla $i = 1$. Jeżeli $i = b$, Apolonia w wyniku otrzyma $s_i + k$, jeżeli $i \neq b$ – przypadkową liczbę. Dla Apolonii obie liczby będą wyglądały równie losowo.
- Apolonia przekazuje Bogusowi t_0 i t_1 . Boguś oblicza $t_b - k$, czyli od właściwego t_i odejmuje swoją początkową wartość k . W wyniku otrzymuje sekret s_b . Boguś nie pozna drugiego sekretu Apolonii – żeby to zrobić, musiałby złamać algorytm RSA.

Bezpieczne obliczenie iloczynu logicznego:

- Protokół ten możemy wykorzystać do rozwiązywania problemu miłosnego, czyli do bezpiecznego obliczenia iloczynu logicznego bitów a i b:

Apolonia	Boguś
$s_0 = 0, s_1 = a$	$b \in \{0, 1\}$
$s_b = a \wedge b$ $\xrightarrow{s_b}$	s_b
	$\xleftarrow{s_b}$
- Dwoma sekretami Apolonii będą 0 i jej tajny bit a (równy jeden, jeżeli Apolonia kocha Bogusia i zero w przeciwnym przypadku).
- Bitem, za pomocą którego Boguś wybierze sobie sekret Apolonii będzie jego tajny bit b.
- Przekazany sekret będzie równy iloczynowi logicznemu a \wedge b
 - Jeżeli Apolonia nie kocha Bogusia i Boguś nie kocha Apolonii, to przekazany zostanie sekret s_0 równy zawsze 0
 - Jeżeli Apolonia nie kocha Bogusia, a Boguś kocha Apolonię, to przekazany zostanie sekret s_1 , równy w tym przypadku 0. (Pamiętamy, że Apolonia niewie, który sekret wybiera sobie Boguś.)
 - Jeżeli Apolonia kocha Bogusia, a Boguś nie kocha Apolonii, wtedy Boguś wybierze sobie sekret $s_0 = 0$ i nie pozna drugiego z jej sekretów.
 - Jeżeli oboje się kochają, wówczas Boguś poprosi o sekret s_1 , który będzie teraz równy 1
- Uzyskanym wynikiem Boguś dzieli się następnie z Apolonią

Wykład 6 – Kryptoanaliza

Typy ataków:

- Atak bez tekstu jawnego (znany szyfrogram, ewentualnie algorytm szyfrujący, język tekstu jawnego, charakterystyczne słowa tekstu jawnego)
- Atak z tekstem jawnym (pary tekstów jawnego i zaszyfrowany)
- Atak z wybranym tekstem jawnym (szyfrogram i odpowiadający mu tekst jawnego)

Znane metody wykorzystywane przy szyfrach historycznych:

- Analiza częstości występowania liter - szyfry podstawieniowe
- Metoda anagramowa - szyfry przestawieniowe (permutacyjne)
- Metoda szukania klucza przez wyczerpujące przeszukiwanie (brute force)

Ataki na szyfry blokowe:

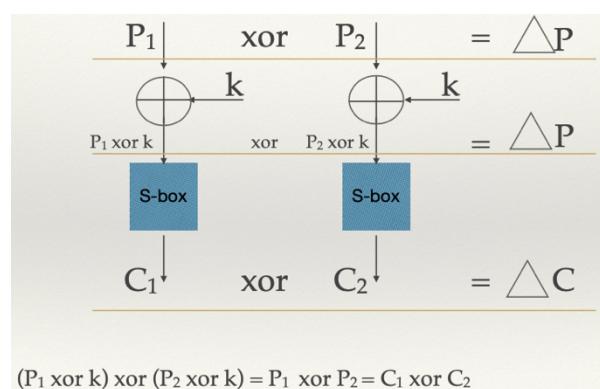
- Ataki standardowe
 - wyczerpujące przeszukiwanie
 - atak słownikowy
 - kryptoanaliza różnicowa
 - kryptoanaliza liniowa
 - ataki algebraiczne
- Ataki na implementacje - side channel attacks
 - czasowe
 - zużycia prądu
 - wstrzykiwania błędów

Atak słownikowy:

- Konstruujemy tablicę $(K, E_K[0])$ dla wszystkich możliwych kluczy (K)
 - sortujemy ją względem drugiego pola - zaszyfrowanego tekstu
- Próbując złamać nowy klucz K (atak ze znanym tekstem jawnym)
 - Wybieramy 0, otrzymany szyfrogram C , przeglądamy tablicę w celu wyszukania odpowiadającego klucza

Kryptoanaliza różnicowa:

- atak z wybranym tekstem jawnym
- stosowana nie tylko do szyfrów blokowych ale także funkcji skrótu
- Główne założenia:
 - atak z wybranym tekstem jawnym (wiele par - szyfrów i tekstów jawnych)
 - Obliczamy różnice $\Delta P = P_1 \text{ xor } P_2$, $\Delta C = C_1 \text{ xor } C_2$
 - Rozkład różnic ΔC dla danych ΔP może dostarczyć informacji o kluczu (pojedynczych bitach klucza)
 - Po znalezieniu kilku/kilkunastu/dziesięciu bitów dla pozostałych bitów klucza zastosuj wyczerpujące przeszukiwanie



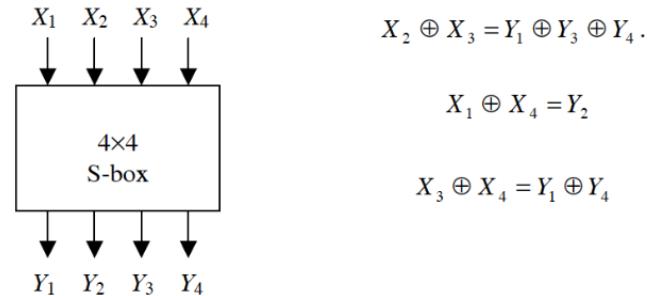
Kryptoanaliza liniowa:

- nie była brana pod uwagę przy projektowaniu algorytmu DES
- wykorzystuje zależności między bitami wejściowymi rundy, bitami klucza oraz wynikami rundy - najważniejszym elementem są formuły aproksymujące liniowo pojedyncze S-bloki
- może być użyta jako:
 - atak ze znanym tekstem jawnym
 - atak z tekstem zaszyfrowanym
- Główne założenie:
 - Podstawowym założeniem jest aproksymacja działania fragmentu szyfru za pomocą wyrażenia liniowego, gdzie liniowość odnosi się do bitowej operacji mod-2 (XOR).

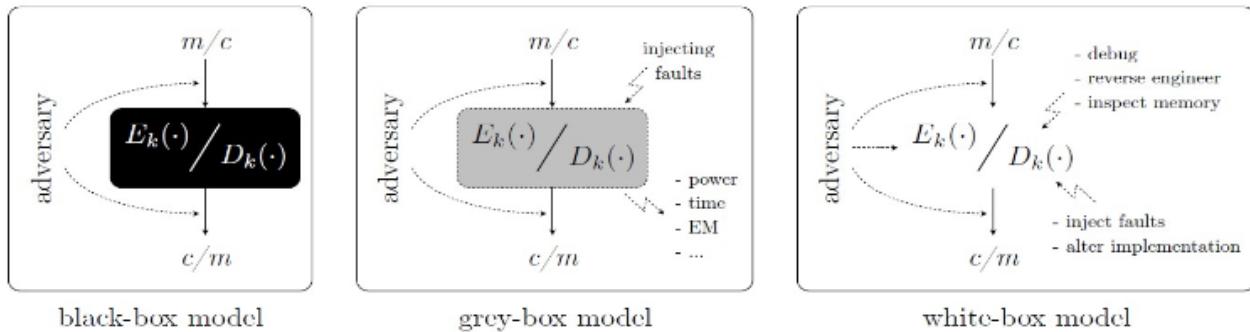
Wyrażenie takie ma następującą formę:

$$X_{i_1} \oplus X_{i_2} \oplus \dots \oplus X_{i_u} \oplus Y_{j_1} \oplus Y_{j_2} \oplus \dots \oplus Y_{j_v} = 0$$

- gdzie X_i reprezentuje i-ty bit wejścia $X=[X_1, X_2, \dots]$ a Y_j reprezentuje j-ty bit wyjścia $Y=[Y_1, Y_2, \dots]$.
- Równanie to przedstawia sumę XOR u bitów wejściowych i v bitów wyjściowych.

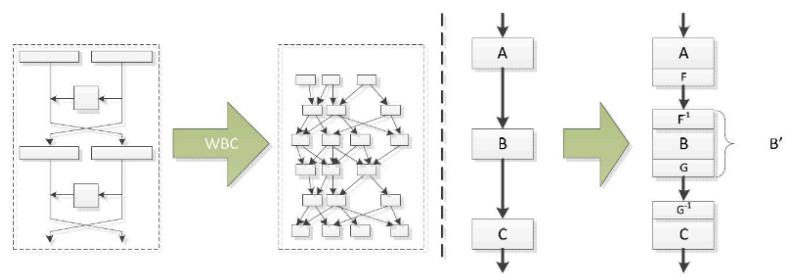


Modele ataków na algorytmy kryptograficzne:



Implementacje white-box:

- Firmy takie jak Microsoft, Apple, Sony i wiele innych ogłosili stosowanie technik white-box
- Problemy:
 - Wydajność - Opracowywane są specjalne algorytmy kryptografii white-box wykorzystujące akceleratory sprzętowe
 - Bezpieczeństwo - Żadna faktyczna implementacja typu white-box nie została złamana, pomimo obiecujących prac teoretycznych



Problemy z szyfrowaniem RSA – szyfrogramy niedeformowalne:

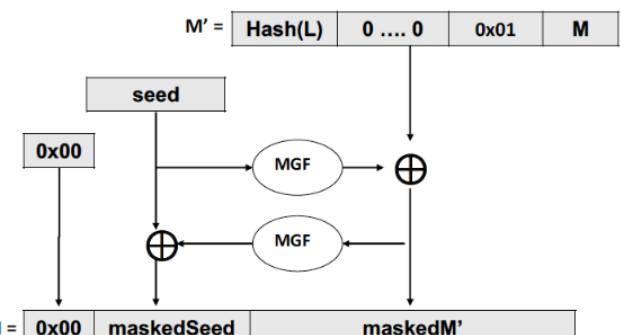
- szyfrowanie podręcznikowe jest deterministyczne
- mając dane dwa szyfrogramy y_1 i y_2 możemy obliczyć szyfrogram $x_1 * x_2$ (sama znajomość szyfrogramów bez znajomości wiadomości jawnych nie powinna na to pozwolić)

PKCS = Public Key Cryptography Standard (opracowane przez RSA Laboratories):

- Zdefiniowano 2 metody kodowania dla szyfrowania RSA
 - RSAES-PKCS1-v1_5
 - RSAES-OAEP
- Nowe aplikacje powinny używać RSAES-OAEP (ponieważ PKCS1-v1_5 jest podatna na atak z wybranym szyfrogramem)
- Obie metody opisują jak dopełniać wiadomość i jak ją przekształcić
- OAEP = Optimal Asymmetric Encryption Padding
- RSAES = RSA Encryption Scheme

RSAES-OAEP-ENCRYPT ((n,e), M, L):

- Opcje:
 - Funkcja skrótu (hLen: rozmiar skrótu w bajtach)
 - MGF: mask generation function - funkcja maski:
 - może wykorzystywać funkcję skrótu
 - różnica: może generować wyjście dowolnej długości
- Wejście:
 - (n, e) publiczny klucz odbiorcy (k: rozmiar n w bajtach)
 - M wiadomość do zaszyfrowania, łańcuch bajtów o długości mLen, gdzie: mLen<=k-2hLen-2
 - L - opcjonalna etykieta połączona z wiadomością, domyślna wartość dla L, jeśli L nie podano, jest to pusty łańcuch
- Wyjście:
 - C tekst zaszyfrowany, łańcuch (bajtów) o długości k
- Uogólniając:
 - łańcuch bajtów wiadomości M jest przekształcany w zakodowaną wiadomość EM k bajtów
 - EM jest zamieniana w liczbę całkowitą
 - Liczba całkowita - reprezentacja EM - jest szyfrowana kluczem (n, e) w liczbę całkowitą c
 - Liczba całkowita c jest przekształcana w łańcuch k-bajtów szyfrogramu C
- Jak działa OAEP:
 - Mamy: M - wiadomość, do zaszyfrowania, generator PRNG oraz dwie funkcje skrótu
 - (maski)
 - dopełnienie zerami: (k – mLen – 2hLen – 2) zerowych bajtów oraz jeden bajt 0x01 (wart. hex)
 - Seed = losowy łańcuch bajtów o długości hLen (innny dla każdej wiadomości M)
 - Zeroowy bajt (oktet) w EM używany jest do detekcji błędów deszyfrowania



Najpopularniejsze algorytmy faktoryzacji:

- MPQS (Multiple Polynomial Quadratic Sieve) - modyfikacja sita kwadratowego,
- ECM (Elliptic Curve Method) - metoda szczególnie skuteczna przy niewielkich czynnikach,
- GNFS (General Number Field Sieve) - najszybszy znany algorytm faktoryzacji

Metody zabezpieczenia RSA:

- dodatkowe warunki:
 - $p-1, q-1, p+1$ i $q+1$ mają mieć duży czynnik pierwszy
 - $|p - q| > 2^{\log_2 p - 100}$
 - $d > N^{0.3}$

Atak Hastada:

- Założmy, że ta sama wiadomość M wysyłana jest do trzech różnych osób, z których każda posługuje się kluczem publicznym postaci $(N_i, 3)$.
- W takiej sytuacji wysyłane są 3 szyfrogramy:
 - $C_1 = M^3 \pmod{N_1}$,
 - $C_2 = M^3 \pmod{N_2}$,
 - $C_3 = M^3 \pmod{N_3}$
- Powyższy układ równań pozwala na odtworzenie wartości $M = M^3 \pmod{(N_1 N_2 N_3)}$.

Atak Franklina-Reitera:

- Metoda ta działa tylko w przypadku wykładnika publicznego o wartości 3.
- Pozwala na złamanie dwóch szyfrogramów $C_1 = M_1^3$ i $C_2 = M_2^3$, które zostały wysłane do tej samej osoby.
- Warunkiem koniecznym przeprowadzenia skutecznej kryptoanalizy jest znajomość liniowej zależności pomiędzy wiadomościami M_1 i M_2 .
- Znając funkcję $f(X) = aX + b \in \mathbb{Z}_N[X]$ ($b \neq 0$), dla której prawdziwa jest zależność $M_2 = f(M_1)$ możemy znaleźć M_1 i M_2

Ataki czasowe:

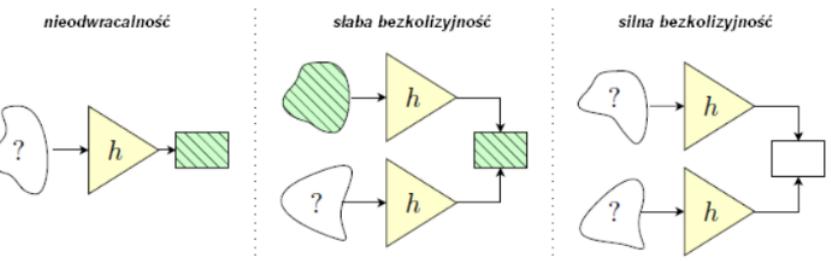
- Czas wykonania poszczególnych instrukcji procesora może być zależny od jej argumentów wejściowych.
- Sytuacja taka ma miejsce w większości współczesnych procesorów, które wyposażone są w całą gamę mechanizmów optymalizujących wykonywanie kodu.
- Pomiar czasu wykonania danej operacji kryptograficznej jest więc w pewnym stopniu skorelowany z przetwarzanymi danymi (w szczególności kluczami kryptograficznymi).

Atak czasowy Kochera:

- Atak pozwala na wyznaczenie wykładnika prywatnego x dla operacji $y^x \pmod{N}$ pod warunkiem, że znane są y i N
 $s_0 = 1$
For $i = 0$ to $n - 1$ **do**
 If $x_i == 1$ **then** $R_i = s_{i-1} y \pmod{N}$
 Else $R_i = s_{i-1}$
 $s_{i+1} = R_i^2 \pmod{N}$
return R_{n-1}
- Technika maskowania losową wartością r^e gdzie e jest wykładnikiem publicznym
- podpisując wiadomość x , wykonujemy $x r^e$ otrzymujemy wtedy:
- $x^d r^{ed} = x^d r \pmod{N}$
- na koniec usuwamy maskę

Funkcje skrótu:

- nieodwracalność:
 - dany jest skrót $h(m)$, wiadomość m jest nieznana. Znalezienie wiadomości m jest obliczeniowo trudne
- słaba bezkolizyjność:
 - dany jest skrót $h(m)$ i odpowiadająca mu wiadomość m . Znalezienie wiadomości $m' \neq m$, takiej że $h(m) = h(m')$, jest obliczeniowo trudne.
- Silna bezkolizyjność
 - obliczeniowo trudne jest znalezienie dowolnej pary różnych wiadomości m' i m , takich że $h(m) = h(m')$.

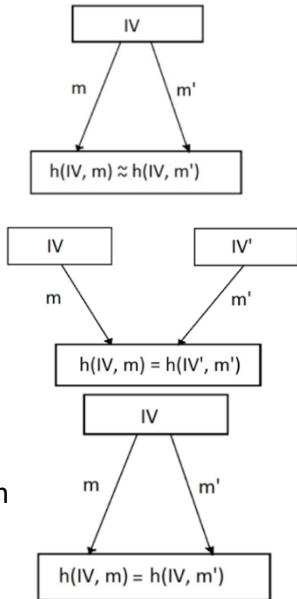


Ataki ogólne na funkcje skrótu:

- wyczerpujące przeszukiwanie (brute force):
 - Ideą ataku brutalnego jest odnalezienie oryginalnej wiadomości przy znajomości jedynie jej skrótu.
 - Atakowana jest jednokierunkowość funkcji skrótu, gdyż intruz próbuje uzyskać przeciwbraz skrótu.
 - Złożoność obliczeniowa wynosi $O(2n)$, gdzie n jest długością skrótu.
 - Złożoność pamięciowa tego ataku jest stała
- atak z użyciem tablic tyczowych
 - Trzeba przygotować tablice (skróty i odpowiadające im wiadomości)
 - Odnalezienie wzorca jest szybsze niż w metodzie czasowej i jest ono rzędu $O(n)$ lub $O(\log n)$ dla tablicy posegregowanej.
 - Złożoność pamięciowa jest znacznie większa i jest ona rzędu $O(2n)$.
- atak słownikowy
 - Polega na siłowym odgadywaniu haseł i w swoim działaniu przypomina atak wyczerpującego przeszukiwania metodą czasową.
 - Główna różnica między nimi polega na sposobie dobierania hasła.
 - Ataki słownikowe bazują tylko na sprawdzeniu podzbioru możliwych wartości, co do którego wiadomo, że jest niewspółmiernie często stosowany przy doborze haseł
- atak urodzinowy
 - Istotą ataku urodzinowego jest znalezienie dowolnej pary wiadomości m i m' takich, że $h(m') = h(m)$. Atak ten oparty jest na paradoksie dnia urodzin, który głosi, że prawdopodobieństwo tego, iż spośród dwudziestu trzech osób, dwie mają urodziny w tym samym dniu wynosi więcej niż jedna druga.
 - Atak ten polega na wygenerowaniu i zapamiętaniu $2^{n/2}$ wiadomości i tego rzędu jest złożoność ataku urodzinowego.

Ataki łańcuchowe:

- ataki wieloblokowe:
 - prawiekolizje:
 - Przy użyciu jednej wartości początkowej IV i dwóch różnych wiadomości m i m' (tj. gdy $m \neq m'$) otrzymujemy skróty różniące się tylko na kilku bitach.
 - Pseudokolizje:
 - Odnalezienie takiej pary wiadomości m i m' (gdzie $m \neq m'$), że przy zastosowaniu dwóch różnych wartości początkowych IV i IV' ($IV \neq IV'$) dadzą ten sam skrót ($h(IV, m) = h(IV', m')$)
 - kolizje przy wybranej wartości początkowej:
 - Po wybraniu dowolnej wartości początkowej IV znajdujemy dla niego dwie różne wiadomości m i m' ($m \neq m'$) dające ten sam skrót $h(IV, m) = h(IV', m')$
- punkty stałe:
 - Punktem stałym jest punkt, w którym wartość odwzorowania na argumentie jest równa temu argumentowi, czyli że $f(x) = x$.
 - Ataki na punkty stałe wykorzystywane są głównie przy eksploataowaniu słabości schematu iteracyjnego MD.
- atak wielokolizyjny,
- atak ze spotkaniem w środku



Narzędzia do przeprowadzania ataków:

- Cain & Abel
- John the Ripper
- Hashcat