Funkcje Skrótu

Paweł Koch 145330, 19.03.2022

1. Zawartość przygotowanego pliku:

MATKOMOJATYJESTESJAKZDROWIEILECIECENICTRZEBATYLKOTENSIEDOWIEKTOC
IESTRACILISTRACILSWOICHBRACILECZWKONCUWROGIZATOZAPLACIOJCZYZNOM
OJACOZDZISIAJMIPOWIESZCZYMACIEKTOBRONICGDYWEKRWITONIESZMYSMYWS
ERCUTWYMPLOMIENROZNIECILIZNOWBYSPOWSTALADUMNANICZYMZNASZYCHSN
OWMIASTOMOJETYSBYLOMIDOMEMICHOCWROGDZISJESTZAKAZDYMZALOMEMJE
SZCZELEPIEJWSZAKTRAFICGONIESZTUKAIZDYCHAKOLEJNAHITLEROWSKAJUCHA
MIASTOMOJEWCIAZBIJETWOJESERCEHUKIEMWYSTRZALOWWBOCZNEJALEJCEIZD
OBYTAKOLEJNAULICAODZNACZAMNALUFIENASTEPNEGOFRYCA

2. Otrzymane wyniki porównań:

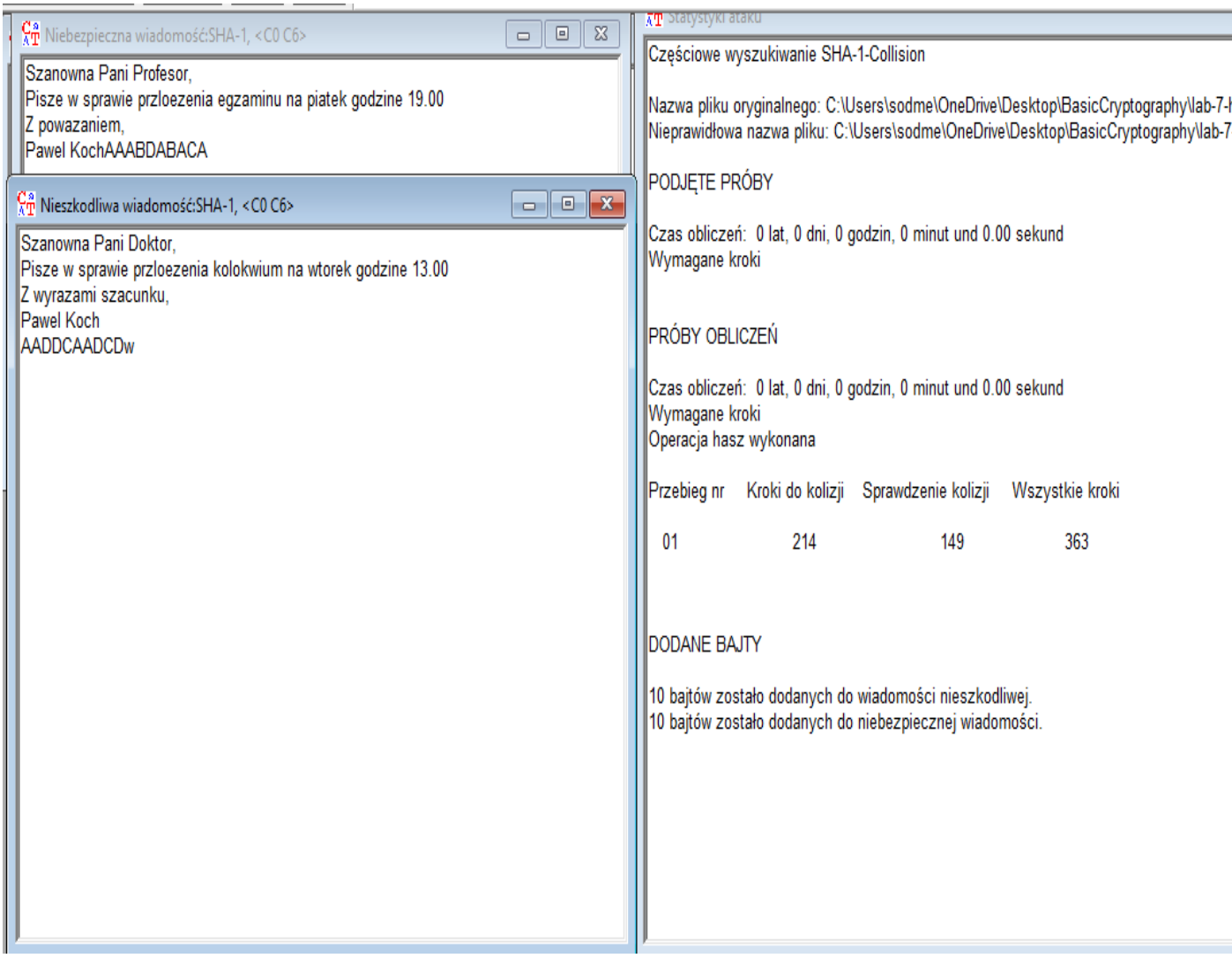
1. SHA-256:

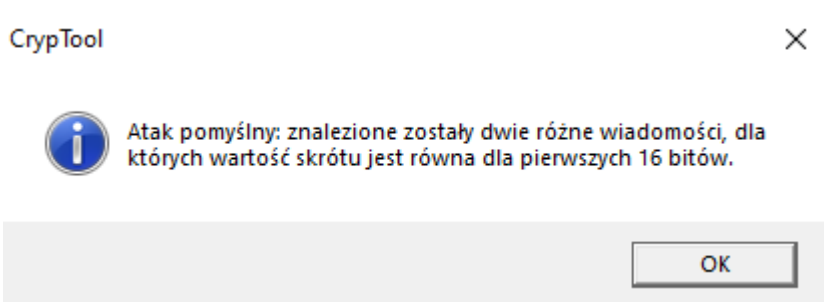
```
Original message: 1100101100101101001100111101111000110110110110110100
0010010000101101011101011101101100001100111000010110100111000111000011
0110110101100001111000110000101100101101111101001100101101001100011111
0001100101100111000011100010111000011000011011110000011101001100101000
0100111000100000111011111010010001001000001110000100011011011111001010
1001011100011011110000101100101101111100000110001101100011100011101111
10100
Modified message: 1101011100001100001100011000011100010110000111100010
1110001110001101111000011100000111100010000111100101100011000001110100
1110101111000110010110010110100100000111011010000111100111101011000100
0110000111000100010011001111011110001001000101110010110000100001111010
1111010110001101110011100111000010110100110110110110100010110001101110
0011000100110111110111100001010000101110001110001000011100000110001000
1000011110111
Diffrence betweeen two hashes is 191 and length of hash is 401
Original message --> Modified Message
0 --> 1 at position 3
1 --> 0 at position 4
0 --> 1 at position 5
1 --> 0 at position 10
1 --> 0 at position 15
1 --> 0 at position 22
1 --> 0 at position 25
1 --> 0 at position 27
1 --> 0 at position 28
0 --> 1 at position 31
1 --> 0 at position 34
1 --> 0 at position 40
1 --> 0 at position 41
0 --> 1 at position 45
1 --> 0 at position 47
0 --> 1 at position 51
0 --> 1 at position 52
1 --> 0 at position 56
0 --> 1 at position 57
0 --> 1 at position 58
```

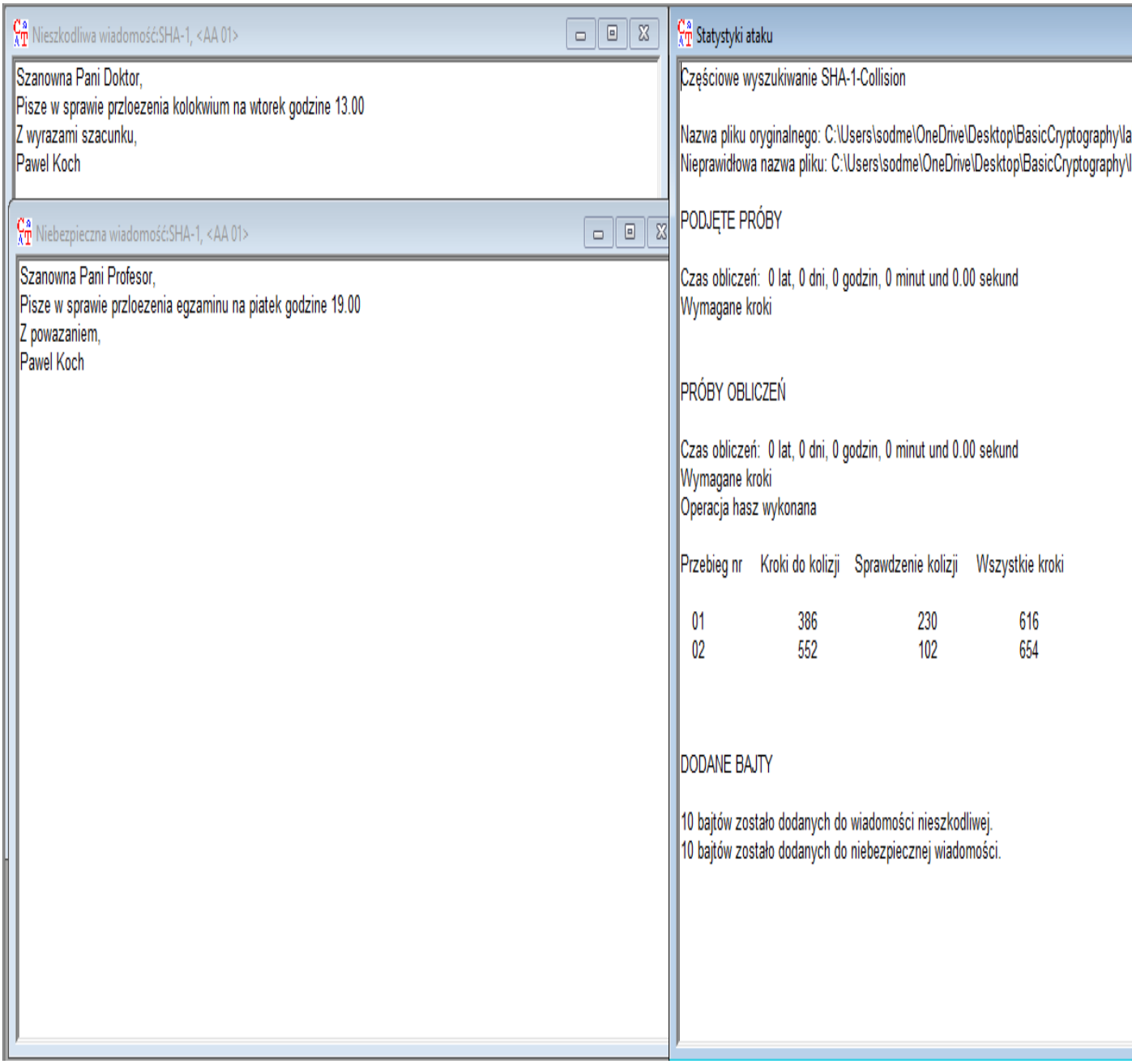
2. SHA-512:

```
Original message: 1000011110001110001110000100010111001011001111001010
0111000110110011111000110100100010011000111100111001110001011100001100
1001000100110100100010011000110000011100101110001000001110001111000100
0111100001000100110100110110110100110101100001111010010000111101001000
0111100110001001000001110110110100110111100001110000011100111101001100
1000011001111100110001001100101100001000101110111100010011000111001110
Modified message: 1100001101111101111000101110010110111111011010000101
0011101111101111000100110111111000110011110110100001111100011011011100
1100111100101100111000010110011110011110000100011011010010001101101100
1111001100001011011111001011100110001011101111101011101111000110110110
1111000110011111001110001100010010000101100011101111101101100111100001
1100111100110000101100101101101100011110001000110111001100010111001110
Diffrence betweeen two hashes is 389 and length of hash is 813
Original message --> Modified Message
0 --> 1 at position 1
1 --> 0 at position 5
1 --> 0 at position 8
0 --> 1 at position 9
0 --> 1 at position 10
0 --> 1 at position 11
1 --> 0 at position 14
0 --> 1 at position 15
0 --> 1 at position 16
0 --> 1 at position 17
1 --> 0 at position 19
1 --> 0 at position 20
0 --> 1 at position 22
0 --> 1 at position 24
0 --> 1 at position 26
1 --> 0 at position 33
0 --> 1 at position 34
0 --> 1 at position 35
0 --> 1 at position 37
1 --> 0 at position 39
0 --> 1 at position 40
0 --> 1 at position 41
1 --> 0 at position 42
1 --> 0 at position 44
1 --> 0 at position 45
```

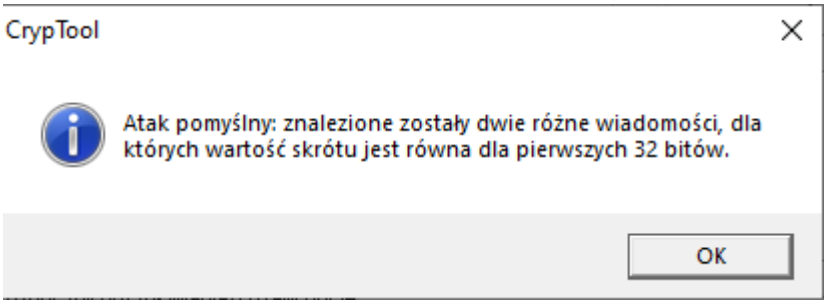
3/4.Szukanie Kolizji







Wersja dla n = 32



Częściowe wyszukiwanie SHA-1-Collision

Nazwa pliku oryginalnego: C:\Users\sodme\OneDrive\Desktop\BasicCryptography\lab-7-hashing\oryginalny.txt
Nieprawidłowa nazwa pliku: C:\Users\sodme\OneDrive\Desktop\BasicCryptography\lab-7-hashing\sfalszowany.txt

PODJĘTE PRÓBY

Czas obliczeń: 0 lat, 0 dni, 0 godzin, 0 minut und 1.72 sekund
Wymagane kroki

PRÓBY OBLICZEŃ

Czas obliczeń: 0 lat, 0 dni, 0 godzin, 0 minut und 0.03 sekund
Wymagane kroki
Operacja hasz wykonana

Przebieg nr	Kroki do kolizji	Sprawdzenie kolizji	Wszystkie kroki
01	17,816	16,282	34,098

DODANE BAJTY

18 bajtów zostało dodanych do wiadomości nieszkodliwej.
18 bajtów zostało dodanych do niebezpiecznej wiadomości.

Wyniki dla SHA-256:

```
Original message: 1101111100111110001101011000001111000100000111
01110110110111110110110010110101110011110100111001110011110101110
11110101111000111000110101100011010000011110001000100111001110110
Modified message: 1101001100111101011101101000010100001110001101
10110110110100010011001111001011011111011010001101101101110011110
00110000111100011110011100101110011110011100111000001110011110001
Diffrence betweeen two hashes is 190 and length of hash is 399
Original message --> Modified Message
1 --> 0 at position 4
1 --> 0 at position 5
1 --> 0 at position 14
0 --> 1 at position 15
0 --> 1 at position 17
0 --> 1 at position 22
1 --> 0 at position 23
0 --> 1 at position 29
1 --> 0 at position 30
1 --> 0 at position 32
1 --> 0 at position 33
0 --> 1 at position 36
0 --> 1 at position 38
0 --> 1 at position 42
1 --> 0 at position 44
1 --> 0 at position 48
1 --> 0 at position 49
0 --> 1 at position 52
0 --> 1 at position 53
1 --> 0 at position 56
1 --> 0 at position 57
0 --> 1 at position 58
1 --> 0 at position 59
1 --> 0 at position 63
0 --> 1 at position 65
```

Wszystkie porównania bitów wykonano za pomocą prostego komparatora:

```python
import readline
import sys

file_to_compare = "lab-7-hashing/zad4.txt"

if __name__ == "__main__":
    with open(file_to_compare) as f:
        content = f.readlines()
        hash_one = content[0].strip().replace(" ", "")
        hash_two = content[1].strip().replace(" ", "")

        hash_one = ''.join(format(ord(x), 'b') for x in hash_one)
        hash_two = ''.join(format(ord(x), 'b') for x in hash_two)

        print(f"Original message: {hash_one}")
        print(f"Modified message: {hash_two}")

        diff = []
        counter = 0

        for i in range(min([len(hash_two), len(hash_one)])):
            if hash_one[i] != hash_two[i]:
                counter += 1
                diff.append((i, hash_one[i], hash_two[i]))

        print(
            f'Diffrence betweeen two hashes is {counter} and length of hash is {len(hash_one)}')
        print("Original message --> Modified Message")

        for x in diff:
            print(f"{x[1]} --> {x[2]} at position {x[0]}")
```

Literatura

Szczególną podgrupą funkcji skrótu są funkcje uznawane za bezpieczne do zastosowań kryptologicznych (jak np. SHA-3). Kryptograficzna funkcja skrótu powinna spełniać kombinację następujących kryteriów, w zależności od zastosowania:

Odporność na kolizje (collision resistance) – brak praktycznej możliwości wygenerowania dwóch dowolnych wiadomości o takim samym skrócie
Odporność na kolizje konkretnych wiadomości (target collision-resistance, preimage resistance) pierwszego i drugiego rzędu – brak praktycznej możliwości wygenerowania wiadomości o takim samym skrócie jak wskazana wiadomość
Jednokierunkowość (one-wayness) – brak możliwości wnioskowania o wiadomości wejściowej na podstawie wartości skrótu. Zmiana dowolnego pojedynczego bitu wiadomości powinna zmieniać średnio połowę bitów skrótu w sposób, który nie jest istotnie podatny na kryptoanalizę różnicową.
Poza wymienionymi w niektórych zastosowaniach od funkcji skrótu wymaga się także:

pseudolosowości (pseudorandomness),

uwierzytelnienia wiadomości (message authentication),
niemożności odróżnienia od losowej wyroczni (indifferentiability from random oracles)[1] – uczynienie niemożliwym do znalezienia dla przeciwnika dwóch wiadomości, dla których wartości funkcji skrótu nieznacznie się różnią.
Ponadto nie powinno być możliwości wywnioskowania żadnych użytecznych informacji na temat wiadomości, mając do dyspozycji tylko jej skrót. Dlatego, funkcje haszujące powinny zachowywać się jak funkcje losowe na ile to możliwe, będąc jednocześnie funkcjami deterministycznymi, łatwymi do obliczenia.

Uznanie funkcji za bezpieczną do zastosowań kryptograficznych opiera się zawsze wyłącznie na domniemanej odporności na znane ataki kryptoanalityczne, nie zaś na matematycznych dowodach gwarantujących niemożność złamania. W szczególności bezpieczna funkcja skrótu musiałaby być funkcją jednokierunkową, a istnienie takich funkcji nie zostało dotychczas dowiedzione. Poważne słabości znaleziono w wielu funkcjach skrótu, które historycznie uchodziły za bezpieczne – m.in. w MD2, MD4, SHA, MD5.

Niekiedy zamiast wyspecjalizowanych funkcji skrótu do realizacji podobnych funkcji bezpieczeństwa stosuje się algorytmy zbudowane na bazie szyfrów blokowych – zwłaszcza w kodach uwierzytelniania wiadomości (MAC).
Część powszechnie stosowanych funkcji skrótu jest podatna na ataki typu length-extension.Ta własność może być wykorzystana do złamania prostych schematów uwierzytelniania bazujących na funkcjach haszujących. Odporność na te ataki była jednym z celów projektowych SHA-3.

Ataki typu denial-of-service

Nawet bezpieczne funkcje skrótu mogą być obiektem ataków kryptograficznych, wykorzystujących miejsce gdzie funkcje te są wykorzystywane w strukturach danych. Wiele struktur danych działa bardzo efektywnie w przeciętnym przypadku, ale słabo w przypadku pesymistycznym. Atakujący może za pomocą niewielkiej ilości specjalnie w tym celu przygotowanych danych przeciążyć system (atak DoS).

Załóżmy, że serwer trzyma pewne dane w tablicy mieszającej – ma k kubełków (buckets), i w każdym trzyma te dane, dla których ostatnie kilka cyfr wartości funkcji skrótu jest równe numerowi kubełka.

Wyszukiwanie wśród danych odbywa się bardzo szybko – jeśli liczba kubełków jest proporcjonalna do ilości danych, to przeciętnie wyszukiwanie odbywa się w czasie stałym.

Jeśli potrafimy jednak znaleźć serię danych, dla których ostatnie cyfry wartości funkcji skrótu używanej przez serwer są identyczne, możemy zmusić serwer do wrzucenia wszystkich danych do tego samego kubełka. Wtedy każde wyszukiwanie będzie wymagać przeszukania wszystkich danych. Już niezbyt duża ilość danych potrafi spowolnić serwer tak bardzo, że nie będzie w stanie wypełniać swojej funkcji. Jest to szczególnie groźne, jeśli serwer ten miał znaczenie dla bezpieczeństwa (IDS, firewall, serwer SSH).

Możliwe zabezpieczenia przed atakami tego typu to między innymi:

używanie struktur danych o lepszej złożoności pesymistycznej, pomimo gorszych wyników przeciętnych (np. drzew czerwono-czarnych zamiast tablic mieszających),
wykrywanie ataków tego typu lub uniemożliwienie ich powstawania przez odrzucanie patologicznych danych,
używanie kryptograficznie bezpiecznych funkcji skrótu dodatkowo wzmocnionych przeciwko takim atakom (np. funkcji skrótu z kluczem jak HMAC-MD5, HMAC-SHA1),
randomizacja skrótów.

Amerykański instytut NIST publikuje zalecenia dotyczące stosowania poszczególnych funkcji skrótu w zależności od pożądanego czasu ochrony informacji. Zgodnie z tymi wytycznymi od 1999 roku nie powinna być stosowana funkcja MD5, zaś funkcja SHA-1 powinna być stosowana co najwyżej do 2010 roku.

Do nowych aplikacji zalecane są funkcje skrótu z rodziny SHA-2, a w przyszłości funkcja SHA-3.

Wnioski

Funkcje skrótu stanowią doskonałą dodatkową warstwę zabezpieczeń jeśli połaczymy je wraz z odpowiednim szyfrowaniem. Doskonale nadaja się do przerzymywania infromacji w nieczytelni dla czlowieka sposób jendoczesnie gwarantując, że przy wprowadzeniu pierwotnej wiadomosci otrzymamy taki sam wynik funkcji. Fakt że funkcje skrótu pozwalaja uzyskac zawsze ciąg o takiej samej długości dodatkowo pozwala używać ich w implementacji wielu struktur danych co jeszcze bardziej zwiększa ich użyteczność.
Nie sa one jednak pozbawione wad, jako że funkcja doskonała nie istnieje (taka, która dla każdego ciągu wejściowego da inny ciąg wyjsciowy) występują ataki kolizyjne majace na celu uzyskanie innym ciągiem naszego wyniku finkcji skrótu, jest to najbardziej znacząca krytpograficznie wada w celu której opracowuje się nowe funkcje skrótu jak np. SHA-3.

Bibliografia

https://web.archive.org/web/20090521001714/http://www.infosec.sdu.edu.cn/uploadfile/papers/How%20to%20Break%20MD5%20and%20Other%20Hash%20Functions.pdf

https://web.archive.org/web/20130731160122/http://homepages.cwi.nl/~pietrzak/publications/FLP08.pdf

https://web.archive.org/web/20130528140757/http://blog.securitystandard.pl/news/342130.html