Podstawy Kryptografii – Sprawozdanie z Laboratorium nr 3 (2.04.2025)

Jakub Bilski (155865)

Informatyka (ST), Semestr VI, Rok 2024/2025

1. Omówienie implementacji

- **Generowanie danych i pomiar prędkości haszowania**
  Funkcja *generate_data(size)* odpowiada za generowanie losowych bloków danych o określonym rozmiarze. Argument size pomnożony przez 1024 * 1024 oblicza liczbę bajtów – np. aby wygenerować 10 MB danych, funkcja tworzy ciąg o długości 10 * 1024 * 1024 bajtów.

  Funkcja *measure_hash_speed(data)* dla każdej funkcji skrótu (MD5, SHA-256, SHA3-512) mierzy czas potrzebny dla obliczenia skrótu dla danego bloku danych. Funkcja przechodzi przez listę algorytmów, rejestruje czas początkowy, wywołuje funkcję skrótu, rejestruje czas końcowy. Czas wykonania operacji to różnica między zebranymi czasami.

- **Test SAC**
  Funkcja *compute_sac()* generuje losowy ciąg bajtów o zadanej długości – dane pierwotne. Następnie obliczany jest oryginalny skrót.

  Funkcja *flip_bit()* zmienia pojedynczy bit w danych, a następnie funkcja *count_bit_differences()* zlicza liczbę różnic w bitach pomiędzy pierwotnym, a oryginalnym skrótem.

  Za pomocą funkcji *count_bit_differences()* porównuje się pierwotny skrót z nowym, zlicza się liczbę zmienionych bitów (poprzez operację XOR między odpowiadającymi sobie bajtami). Wynik tej operacji to liczba zmienionych bitów w wyniku modyfikacji jednego bitu w pierwotnym bloku danych.

  Całkowita liczba zmienionych bitów jest sumowana dla wszystkich możliwych pojedynczych zmian, a następnie obliczana jest średnia liczba zmienionych bitów poprzez podzielenie sumy zmienionych bitów przez liczbę bitów w danych wejściowych.

Aby obliczyć skuteczność testu SAC, średnia liczba zmienionych bitów jest dzielona przez długość skrótu w bitach – wynik podawany jest w procentach. Oczekiwana wartość – 50% (prawdopodobieństwo 0,5).

- **Test kolizji prefiksu**
  Funkcja *test_prefix_collisions()* generuje losowy ciąg bajtów *m* o zadanym rozmiarze. Na podstawie uzyskanego ciągu bajtów *m* obliczamy skrót *h_m* przy użyciu funkcji SHA-256.

  Dla liczby prób *num_trials* generujemy kolejne losowe ciągi bajtów *m_prime.* Na podstawie każdego nowego ciągu oblicza się nowy skrót *h_m_prime*. Z nowego skrótu porównujemy pierwsze *n* bitów z *n* bitów ze skrótu *h_m*.

  Za pomocą *hash_prefix_equal()* obliczamy liczbę bajtów potrzebnych do pokrycia *n* bitów. Ponieważ liczba porównywanych bitów nie musi być podzielna przez 8, ostatni bajt jest maskowany tak, aby porównać tylko interesujące nas bity. W przypadku, gdy porównane prefiksy są takie same, to inkrementujemy licznik dla prefiksów o długości *n*.

  Przy założeniu, że mamy do czynienia z idealnie losowym rozkładem bitów, prawdopodobieństwo, że dwa skróty mają identyczny *n*-bitowy prefiks, wynosi 1/2^*n*. Dla liczby prób *num_trails* prawdopodobieństwo to wynosi *num_trails*/2^*n*.

2. Rola soli w tworzeniu skrótów

Sól jest (losowym) ciągiem danych, który dodaje się do pierwotnego tekstu jeszcze przed jego zahaszowaniem. Wśród najważniejszych ról soli można wymienić:

- **Przekształcenie skrótu w unikalny –** w przypadku, gdy napotkamy na takie samo hasła, dodanie do nich różnych soli sprawi, że wynikowy hash będzie inny.
- **Ochrona przed atakami słownikowymi –** jeśli atak będzie opierać się na sprawdzaniu najpopularniejszych haseł (np. password, 123456, qwerty), to może powoływać się na gotowe hashe wygenerowane na ich podstawie. Dodanie soli zwiększa szanse na to, że atakujący nie znajdzie naszego skrótu w zbiorze gotowych, wcześniej znanych hashów.
- **Zwiększenie entropii –** zwiększenie złożoności danych wejściowych, co utrudnia możliwość odtworzenia pierwotnych danych z samego hasha.

3. Funkcja MD5 – bezpieczeństwo w 2025 roku

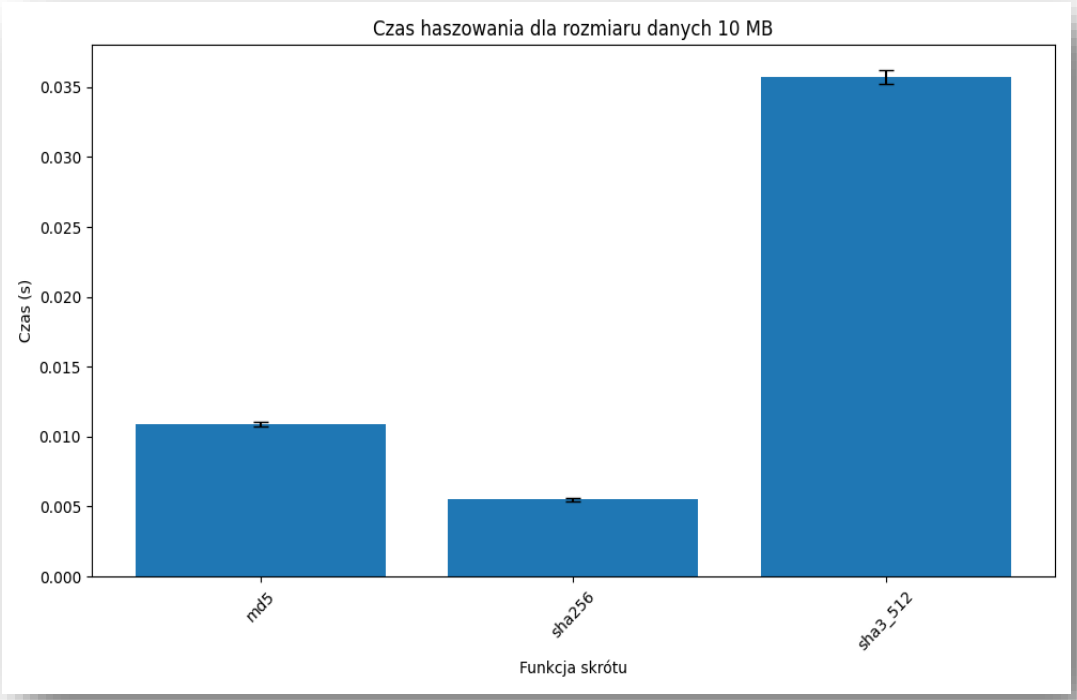
Czy funkcję MD5 można uznać za bezpieczną? Czy dotychczas zostały znalezione dla niej jakiekolwiek kolizje?

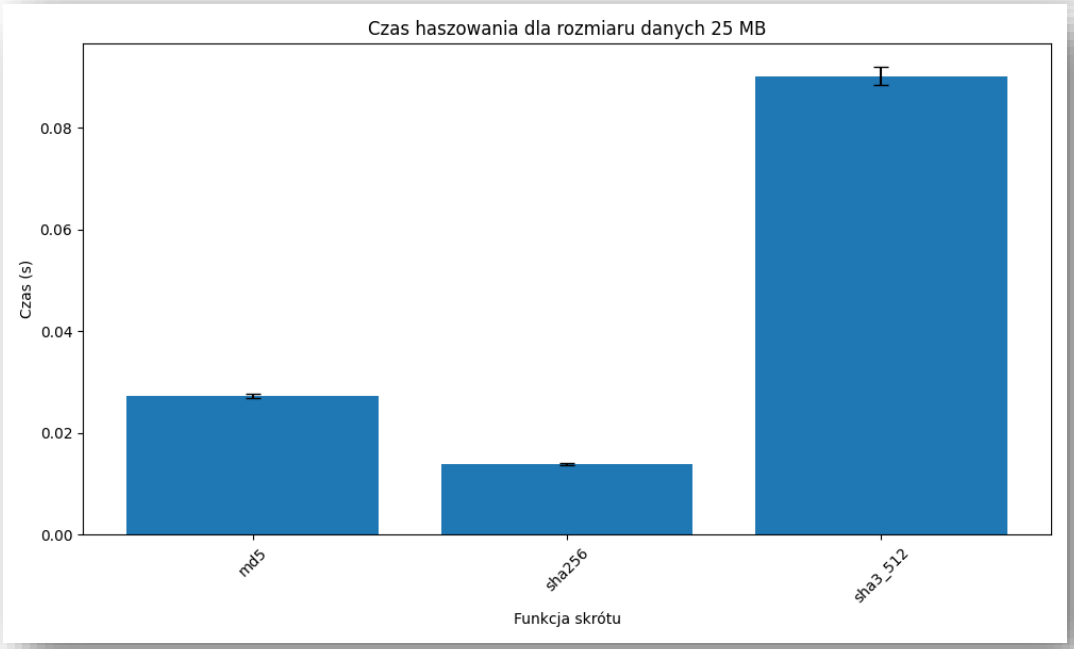
Funkcja skrótu kiedyś cieszyła się bardzo dużą popularnością, jednak z upływem lat udowodniono jej poważne luki. Obecnie uchodzi za niebezpieczną, nie zaleca się jej wykorzystywania.

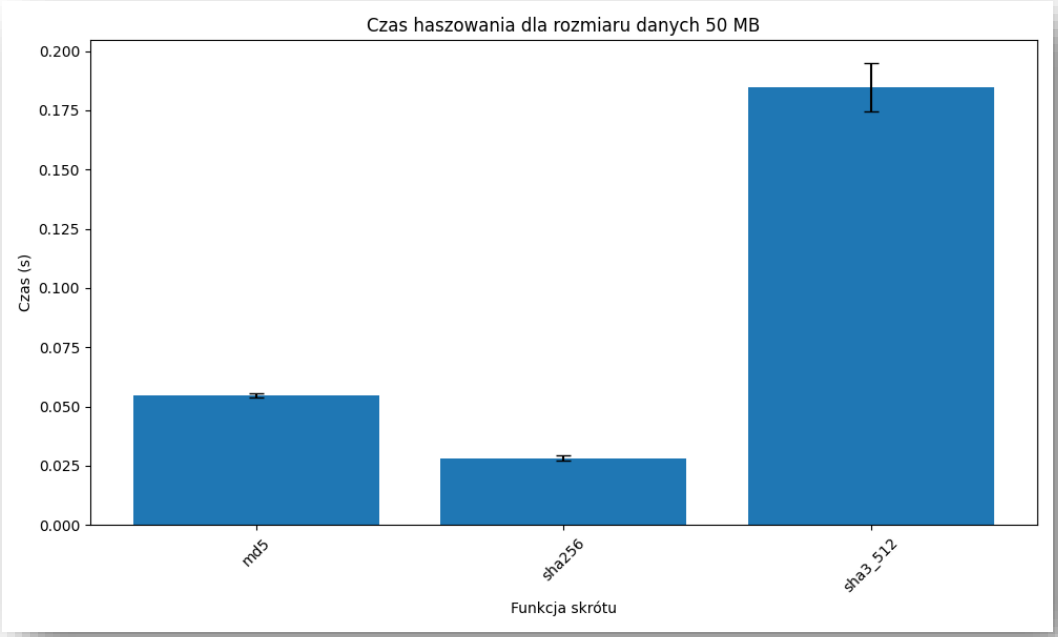
Pierwszą osobą, która wykazała braki w bezpieczeństwie algorytmu, był niemiecki kryptograf Hans Dobbertin w 1996 roku. Osiem lat później chińscy naukowcy przedstawili praktyczną metodę generowania kolizji dla MD5 – dwa różne dane wejściowe mogą dać identyczny hash. Od tamtego czasu udało się już wielokrotnie przeprowadzić ataki kolizyjne, przez co świat odszedł od funkcji MD5 na rzecz nowszych, bezpieczniejszych i bardziej odpornych na kolizje algorytmów. Uznano, że jest ona nieodpowiednia do zastosowań wymagających wysokiego poziomu bezpieczeństwa.

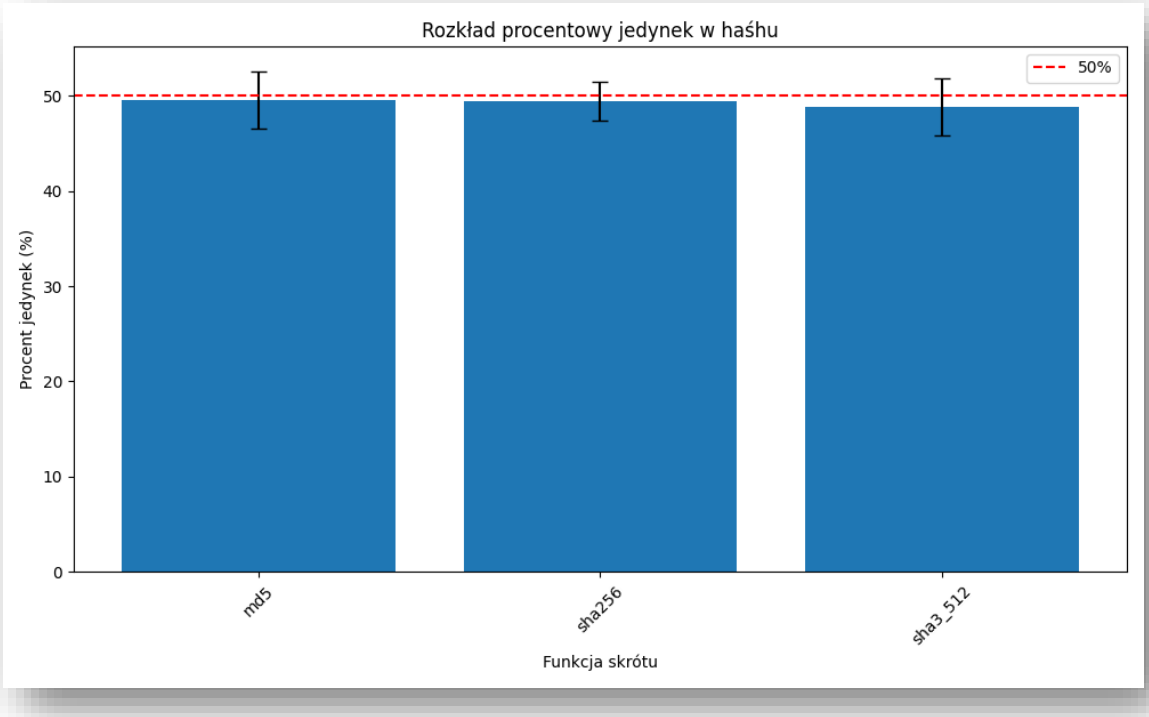
Co z tego, że MD5 jest dość szybką funkcją, skoro tak naprawdę jest to zachęta do przeprowadzenia ataków metodą brute-force. Prawdopodobieństwo zakończenia ich sukcesem jest o wiele wyższe niż w przypadku innych funkcji skrótu.

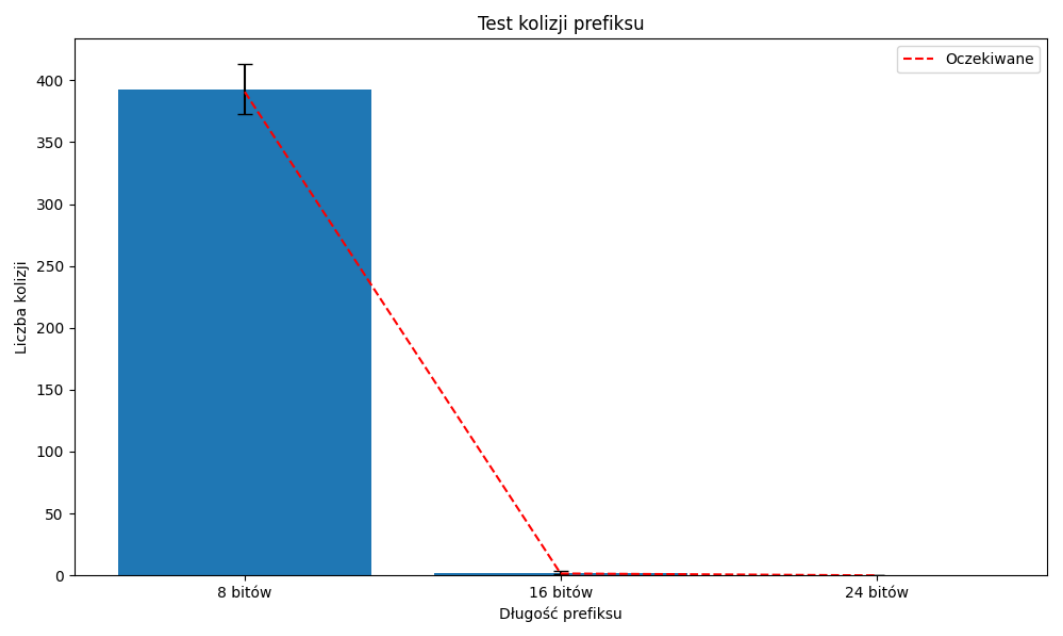
4. Zestawienie uzyskanych wyników oraz wnioski

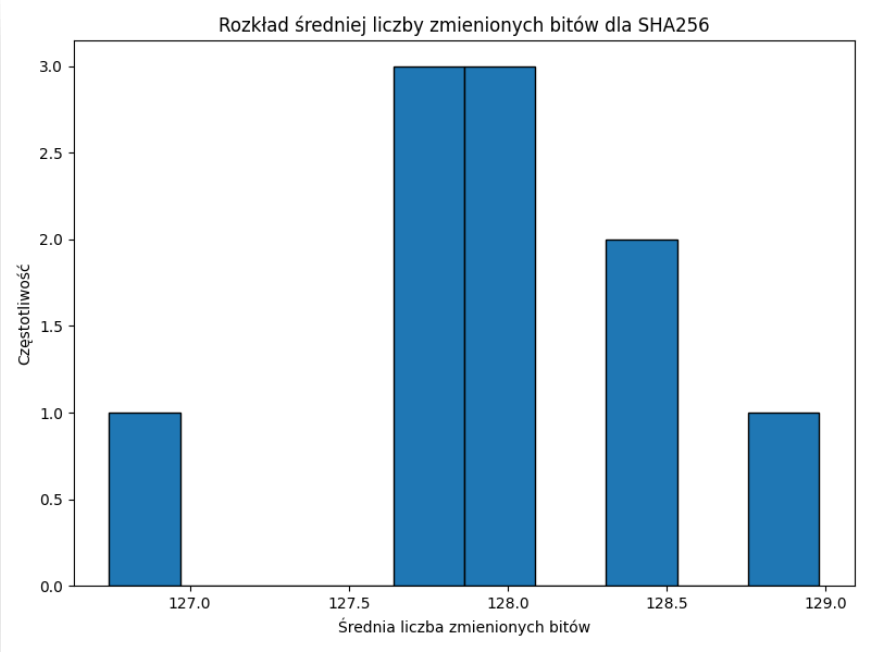





Czas haszowania dla rozmiaru danych 50 MB
Czas (s)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
md5
sha256
sha3_512
Funkcja skrótu


Rozkład procentowy jedynek w haśhu
50%
Procent jedynek (%)
50
40
30
20
10
0
md5
sha256
sha3_512
Funkcja skrótu


Test kolizji prefiksu
Oczekiwane
Liczba kolizji
0
50
100
150
200
250
300
350
400
8 bitów
16 bitów
24 bitów
Długość prefiksu


Rozkład średniej liczby zmienionych bitów dla SHA256
Częstotliwość
0.0
0.5
1.0
1.5
2.0
2.5
3.0
127.0
127.5
128.0
128.5
129.0
Średnia liczba zmienionych bitów

Prędkość haszowania:

- Choć intuicyjnie moglibyśmy się spodziewać, że MD5 – będąc starszym i „lżejszym” algorytmem – będzie szybszy, wyniki jednoznacznie pokazują, że SHA256 osiąga nawet około dwukrotnie lepsze czasy. Dla przykładu, przy 10 MB danych MD5 wykonuje operację w około 0,01 s, podczas gdy SHA256 w około 0,005 s. Podobna proporcja występuje przy większych rozmiarach.
- SHA3_512 oferuje znacznie lepszą odporność na ataki i większą długość wyjściowego hashu, co przekłada się na wyższy poziom bezpieczeństwa. Jednak za poprawę bezpieczeństwa płacimy prędkością algorytmu.

Test SAC:

- Sumując różnice dla wszystkich możliwych pojedynczych zmian, a następnie dzieląc przez liczbę bitów w wejściu, otrzymujemy średnią liczbę zmienionych bitów. Dla funkcji skrótu o długości 256 bitów idealny wynik to około 128 bitów, czyli 50% zmiany.
- Dla SHA256 wynik oscylujący wokół 50% zmienionych bitów (127,35 z 256, czyli ok. 49,745%) jest niemal idealny.
- Efekt lawinowy jest kluczowy, gdyż zapewnia, że nawet minimalna zmiana wejścia (np. zmiana jednego bitu) generuje całkowicie inny hash. W praktyce oznacza to, że nie istnieje korelacja między drobnymi modyfikacjami wejścia a zmianami w hashu, co utrudnia analizę i ataki.
- Powtarzanie testu dla wielu losowych danych (co pozwala zbierać statystyki) daje pewność, że efekt lawinowy nie jest przypadkowy – systematycznie osiągamy wartości bliskie 50%.

Test kolizji prefiksu:

- Dla prefiksu $n = 8$ i liczby prób *num_trails* = 100000 oczekiwana liczba kolizji wynosi ok. $100000/256 \approx 391$.
- Dla $n = 16$, $100000/65536 \approx 1.5$
- Dla $n = 24$, oczekiwana liczba kolizji jest niemal zerowa ($\approx 0{,}006$)
- Wyniki testu pokazują, że dla 8-bitowego prefiksu liczba kolizji oscyluje w pobliżu wartości teoretycznej (391). To oznacza, że funkcja skrótu SHA256 generuje wyniki o wysokiej entropii i spełnia model losowego rozkładu.

- Dla 16-bitowego i 24-bitowego prefiksu kolizje praktycznie nie występują, co jest zgodne z bardzo niskim prawdopodobieństwem ich wystąpienia.
- Test kolizji prefiksu nie mierzy pełnej kolizji całego hashu, lecz tylko fragmentu. Jednak fakt, że już na poziomie 8 bitów wyniki pokrywają się z teoretycznymi przewidywaniami, świadczy o równomiernym rozkładzie bitów.
- Analiza kolizji prefiksu dostarcza informacji o tym, jak dobrze algorytm radzi sobie z utrzymaniem unikalności skrótu w zależności od długości używanego prefiksu.
- Wyniki pokazują, że choć dla krótkich prefiksów liczba kolizji może wydawać się wysoka, to jest to zgodne z teorią – co oznacza, że bezpieczeństwo funkcji nie wynika jedynie z braku kolizji całego hashu, ale z ogólnej losowości rozkładu bitów.

Rozkład bitów w hashu:

- Wszystkie testowane algorytmy (MD5, SHA256, SHA3_512) wykazują wartości bliskie 50%, choć z niewielkimi odchyleniami. Jest to bardzo ważna kwestia, ponieważ nierównomierny rozkład mógłby wskazywać na możliwe opcje ataku lub nieodpowiednie rozproszenie informacji.