

# Sprawozdanie – Funkcje skrótu

Mateusz Kreczmer 151736

## 1. Screenshot z aplikacji.

```
Enter text: Kot
MD5 hash: c0d03d2d3e717da54ffdfc8a76c0f089
SHA1 hash: a0e5cd812455e04d6e33646cd8dc17e05b674231
SHA224 hash: bacfd6a442d02b096f13baa3e2e11b1ca81e702351f0d3289b18592d
SHA256 hash: aedaa3e798149ebaec99435ea67f2f1fc8b5cd2f3b039b885bdf8c04678c03
SHA384 hash: 2c4cd621cf134cc6d04c76012b3f8c06247463c95ce9f6158a8b2218f2e9ab323e111551c494e8b92bb95955df3913d
SHA512 hash: 71062884a3882e5442aa9acaff2b570c3470431f60e868a4178cbd0460f71ac0d5a679f4647003d5b683a7c9091a49b1075c72dc63bbbc9aa461c3af9129ddd9
SHA3_224 hash: b6dd6f2e4082800cea403f7e387f80e2be1ee0ad6860ea5f6843dd45
SHA3_256 hash: 16c6f78ba37ae968b2602249278aad82aea653662b3e9583d598030f0fef5c4d
SHA3_384 hash: 8e24b4f5a8c62208ee9161f4ff8bcbfd9d0aced7a0ee8091ff74abf3f2058112c48d10e4fc6b4b037055579789ab18404
SHA3_512 hash: 4e8c7f92a8e50c690ebfbc01aca3e91d29fae439427aa979271d159f4a0bc4f741b3bab6144ecf36cacf21e23aa1720ce952fba1b8836dd7c819e4c6d73604b4a
Algorithm: MD5, Average Time: 0.0000021 s, Average Length: 32.0 chars
Algorithm: SHA1, Average Time: 0.0000016 s, Average Length: 40.0 chars
Algorithm: SHA224, Average Time: 0.0000018 s, Average Length: 56.0 chars
Algorithm: SHA256, Average Time: 0.0000018 s, Average Length: 64.0 chars
Algorithm: SHA384, Average Time: 0.0000028 s, Average Length: 96.0 chars
Algorithm: SHA512, Average Time: 0.0000028 s, Average Length: 128.0 chars
Algorithm: SHA3_224, Average Time: 0.0000035 s, Average Length: 56.0 chars
Algorithm: SHA3_256, Average Time: 0.0000038 s, Average Length: 64.0 chars
Algorithm: SHA3_384, Average Time: 0.0000043 s, Average Length: 96.0 chars
Algorithm: SHA3_512, Average Time: 0.0000053 s, Average Length: 128.0 chars
Found collision for md5 on first 12 bits after 164 tests
Enter second text: Kou
For MD5 bits change prob equals 0.5390625
For SHA1 bits change prob equals 0.50625
For SHA224 bits change prob equals 0.48660714285714285
For SHA256 bits change prob equals 0.53125
For SHA384 bits change prob equals 0.5260416666666666
For SHA512 bits change prob equals 0.439453125
For SHA3_224 bits change prob equals 0.49107142857142855
For SHA3_256 bits change prob equals 0.5234375
For SHA3_384 bits change prob equals 0.515625
For SHA3_512 bits change prob equals 0.513671875
```

## 2. Omówienie sposobu implementacji.

Implementacja tego kodu obejmuje kilka funkcji. Funkcja **generate\_hash** przyjmuje tekst i algorytm jako argumenty i zwraca wynik skrótu dla danego tekstu za pomocą wybranego algorytmu. Do hashowania tekstów odpowiednimi algorytmami użyłem biblioteki **hashlib**.

Funkcja **calculate\_average\_speed\_and\_length** oblicza średni czas wygenerowania skrótu i średnią długość skrótu dla zestawu danych tekstowych.

Funkcja **collision\_on\_first\_12\_bits** szuka kolizji na pierwszych 12 bitach wyniku skrótu dla wybranego algorytmu. Badam występowanie kolizji poprzez generowanie losowych 8-znakowych tekstów, aż do momentu wygenerowania takiego, który koliduje z wcześniej już wygenerowanym.

Funkcja **check\_bit\_flip\_probability** oblicza prawdopodobieństwo zmiany bitu między dwoma tekstami dla danego algorytmu skrótu.

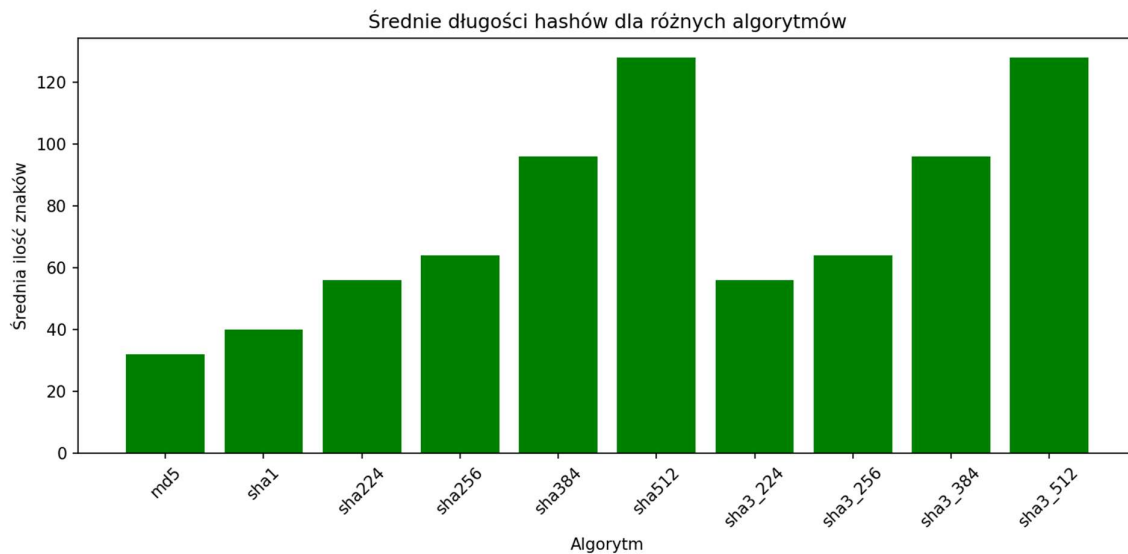
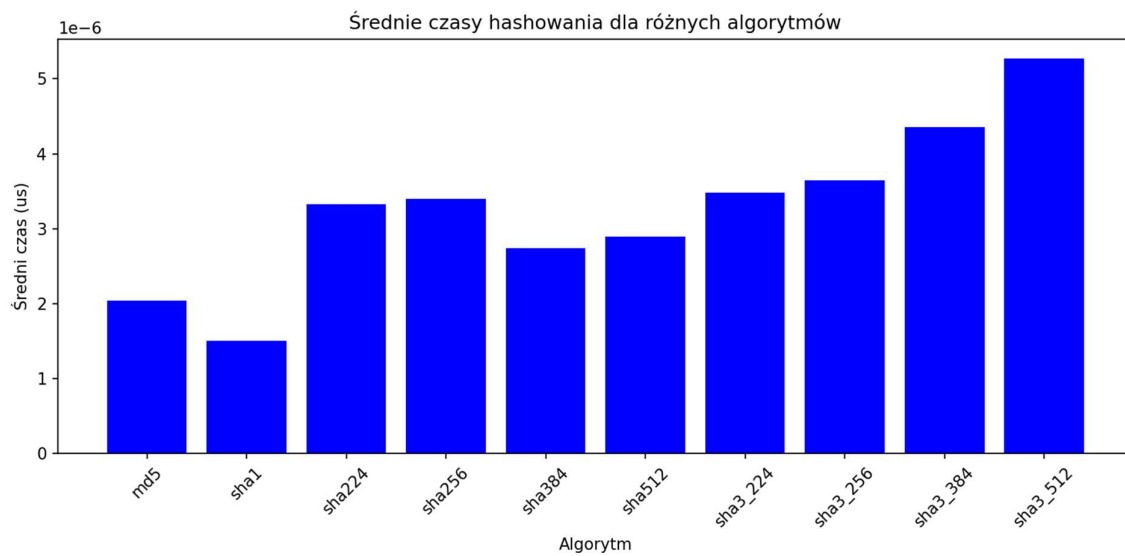
## 3. Określenie roli soli w tworzeniu skrótów.

Rola soli w tworzeniu skrótów polega na dodaniu losowego ciągu danych (soli) do tekstu przed obliczeniem skrótu. Sól zapobiega atakom typu "rainbow table", gdzie wcześniej obliczone skróty są przechowywane w tabelach dla szybkiego odwzorowania skrótów na ich pierwotne wartości. Dodanie soli powoduje, że każde wygenerowane hasło ma unikalny skrót, nawet jeśli są to te same hasła.

4. Na podstawie powszechnie dostępnych źródeł odpowiedz na pytanie – czy funkcję MD5 można uznać za bezpieczną? Czy dotychczas zostały znalezione dla niej jakiegokolwiek kolizje?

Na podstawie powszechnie dostępnych źródeł, funkcję MD5 nie można uznać za bezpieczną. Dotychczas znaleziono szereg ataków kryptograficznych, które wykazują jej słabość. Przykładowo, ataki takie jak "collision attack" umożliwiają wygenerowanie dwóch różnych wiadomości, które mają taki sam skrót MD5. W przypadku zastosowań wymagających wysokiego poziomu bezpieczeństwa, zaleca się wybieranie bardziej bezpiecznych algorytmów skrótu, takich jak SHA-2 lub SHA-3 ponad MD5.

5. Zestawienie uzyskanych wyników wraz ze stosownymi wnioskami.



Na podstawie uzyskanych wyników można zauważyć, że różne algorytmy skrótu mają różne czasy wykonania oraz generują skróty o różnych długościach. Na przykład, algorytm MD5 wykazuje krótsze czasy wykonania niż algorytmy SHA-3, ale generuje skróty o stałej długości 32 znaków. Z drugiej strony, algorytmy SHA-3 mają dłuższe czasy wykonania, ale generują skróty o zmiennych długościach, w zależności od wariantu. Wniosek wynika stąd, że wybór algorytmu skrótu powinien być dokładnie przemyślany, z uwzględnieniem równowagi między czasem wykonania a bezpieczeństwem skrótu oraz wymaganą długością skrótu. Na przykład, w przypadku zastosowań, gdzie ważne są krótkie czasy obliczeń, algorytm MD5 może być wystarczający, podczas gdy w bardziej wymagających sytuacjach zalecane jest stosowanie bardziej bezpiecznych algorytmów, takich jak SHA-3, kosztem wydłużenia czasu obliczeń.