

Funkcje skrótu - Hash functions

Heorhi Zakharkevich
153992

1. Screenshot z aplikacji

To jest tylko 1st zadanie, pozostałe można znaleźć po uruchomieniu samej aplikacji (plik main.py)

```
Write here your phrase -> kot
Phrase: kot
Binary representation: 1101011 1101111 1110100

>===== Zadanie 1. <=====
Hashes:

MD5: a986d9ee785f7b5fdd68bb5b86ee70e0
SHA1: 9eee480a611fbc7a5cd95f2076c26a3084083aef
SHA224: a2de5755e95c11ab8e0e121d43c167760553fb2e1def24269ea50fba
SHA256: b6a5ff9e10883d2329be9ef74cdf1d78ee546f719362fb4325040928a386a520
SHA384: e6961eb5cb8cb85a75fe6faacbec3ee611bd78b60f715310320817eabc3ab3ebb3a05ec63d52be07da071caff74636cd
SHA512: 17efdbe2be21fe1cb9e7c8b6c87cc59835651f9c2c6937e39b4dedd3acfb60e44e2eafda2739a42fe9496c6c16b0ff86dabdf49c85385e72806d7842678567f0
SHA3_224: de3b8025e26752f5631ad02255fb5a51b36680e90de75e07d8c665ad
SHA3_256: e54508d3d90dabdcce7e6081f780edcb872f33a60aac494e0aaa982174780bdb
SHA3_384: 99a32ebc3118481fbb4772deafb244e69a072d30026eb0dd5643c4bdb4a44aaa626b40063a97595bf9d1187ff0c4e3c
SHA3_512: a33b9c1eea4eb3bcc5fa977c6a09fd4ff8b7eb60178515b5edda6c43fc7ded235d59f00b4f0451163b943f55bca75b03b50fa80340e5a307374b8b4768a0aed9
```

2. Omówienie sposobu implementacji

Kod zawiera kilka funkcji:

- **inputText()** - prosi user'a o wprowadzeniu frazy i zwraca ją
- **textToBin()** - konwertuje tekst na reprezentację binarną
- **getAllHashes()** - wyświetla wartości hash'ów dla każdej funkcji którą przyjmuje
- **generateHash()** - generuje wartość hash'a dla danego zakodowanego tekstu i funkcji hashującej za pomocą biblioteki hashlib
- **compareHashFunctions()** - porównuje (wyświetla) szybkość działania i długość wyjścia różnych funkcji hashujących
- **findHashMD5()** - przyjmuje słowo i ścieżkę do pliku, a następnie wyszukuje to słowo w pliku, porównując jego wartość hasha MD5 (nie wykorzystałem w aplikacji)
- **checkHashCollisions()** - sprawdza kolizje na pierwszych 12 bitach hasha dla danej funkcji hashującej i wyświetla znalezione kolizje
- **checkSAC()** - sprawdza spełnienie Ścisłego Kryterium Przepływu (Strict Avalanche Criteria - SAC) dla danej funkcji hashującej i słowa.
- **main()** - służy jako punkt wejścia, wywołując pozostałe funkcje i demonstrując ich użycie.

3. Określenie roli soli w tworzeniu skrótów

W swojej aplikacji nie wykorzystałem koncepcje dodawania soli. W prostych słowach sól w kryptografii to dodatkowy losowy ciąg znaków dodawany do danych wejściowych przed zhashowaniem. Jej celem jest utrudnienie ataków słownikowych i ataków z wykorzystaniem preobliczonych tabel hashów, poprzez wprowadzenie unikalnych soli dla każdego hasła lub danych wejściowych.

4. Odpowiedź oraz jej uzasadnienie na pytanie postawione w pkt. 4

W ciągu ostatnich kilkunastu lat opublikowano wiele efektywnych ataków na MD5, umożliwiających znalezienie kolizji, czyli różnych plików dających ten sam skrót MD5. Ze względu na te podatności, MD5 nie jest już rekomendowana do zastosowań wymagających wysokiego poziomu bezpieczeństwa.

5. Zestawienie uzyskanych wyników

Wszystko można odnaleźć po uruchomieniu aplikacji