

## Podstawy kryptografii – funkcje skrótu

### 1. Screenshot z aplikacji:

```

Input: Lorem ipsum dolor sit amet, consectetur adipiscing elit.
MD5: 35899082e51edf667f14477ac000cbba (Time: 0.000063 sec)
SHA-1: e7505beb754bed863e3885f73e3bb686bdd7f8c (Time: 0.000036 sec)
SHA-256: a58dd8680234c1f8cc2ef2b325a43733605a7f16f288e072de8eae81fd8d6433 (Time: 0.000032 sec)
SHA3-512: efb580d4e7145bd4dd10153624b747d21d09b79c0a2c708de09c19dcd6901a0c34d78ad0e7b7c64f46b7f3ab85aeca7d0f718e9cf38089ad1c9cc05c45ae7ee (Time: 0.000022 sec)

Input: Hello World!
MD5: ed076287532e86365e841e92bfc50d8c (Time: 0.000007 sec)
SHA-1: 2ef7bde608ce5404e97d5f042f95f89f1c232871 (Time: 0.000005 sec)
SHA-256: 7f83b1657ff1fc53b92dc18148a1d65dfc2d4b1fa3d677284add200126d9069 (Time: 0.000005 sec)
SHA3-512: 32400b5e89822de254e8d5d94252c52bdcb27a3562ca593e980364d9848b8041b98eabe16c1a6797484941d2376864a1b0e248b0f7af8b1555a778c336a5bf48 (Time: 0.000007 sec)

Input: 1234567890
MD5: e807f1fcf82d132f9bb018ca6738a19f (Time: 0.000006 sec)
SHA-1: 01b307acba4f54f55aafc33bb06bbbf6ca803e9a (Time: 0.000005 sec)
SHA-256: c775e7b757ede630cd0aa1113bd102661ab38829ca52a6422ab782862f268646 (Time: 0.000004 sec)
SHA3-512: 36dde7d288a2166a651d51ec6ded9e70e72cf6b366293d6f513c75393c57d6f33b949879b9d5e7f7c21cd8c02ede75e74fc54ea15bd043b4df008533fc68ae69 (Time: 0.000006 sec)

Input: gepard
MD5: c5d7e9d3a39a34b1e5f41d064254d774 (Time: 0.000005 sec)
SHA-1: 3e8bd9c9bb32f71e6636e4f2d24a9b019247e29c (Time: 0.000004 sec)
SHA-256: 29d1ef6d9b0a5045fb118ff48275184874f7da14c9585dc266cd5a065db93570 (Time: 0.000004 sec)
SHA3-512: 73aa64bcf8cc31cc3d5ed2d3de40940988106d66bef9ea7736b563077396f3cb2edd8c6eb16b18d7b7fd4b2e6f471550bee48acc78bd4c6410b48113f428df (Time: 0.000006 sec)

Input: a
MD5: 0cc175b9c0f1b6a831c399e269772661 (Time: 0.000005 sec)
SHA-1: 86f7e437faa5a7fce15d1ddcb9eaaea377667b8 (Time: 0.000004 sec)
SHA-256: ca978112ca1bbdcafac231b39a23dc4da786eff8147c4e72b9807785afee48bb (Time: 0.000004 sec)
SHA3-512: 697f2d856172cb8309d6bb97dac4de344b549d4dee61edfb4962d8698b7fa803ff4f93ff24393586e28b5b957ac3d1d369420ce53332712f997bd336d09ab02a (Time: 0.000005 sec)

MD5 hash for 'gprd': 63d9994c4eb8ad51c6107a95de2b78bd5
Collision probability in first 12 bits: 0.036
SAC score for SHA-256: 0.4844

```

### 2. Sposób implementacji

Aplikacja została zaimplementowana w języku Python, przy użyciu modułu „hashlib”, który zapewnia dostęp do różnych algorytmów skrótu. W mojej implementacji porównuję czasy działania następujących algorytmów: MD5, SHA-1, SHA-256 oraz SHA3-512. Badam również szansę wystąpienia kolizji na pierwszych 12 bitach oraz losowość wyjścia funkcji skrótu SHA-256.

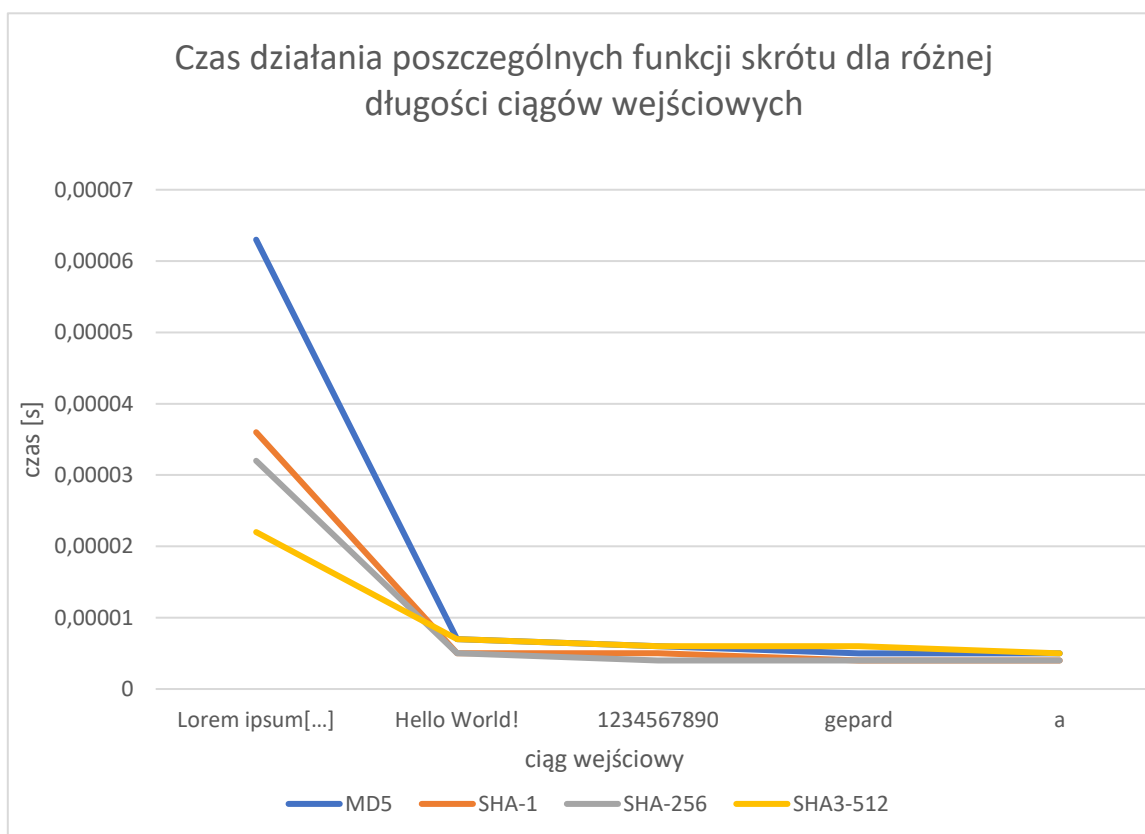
### 3. Rola soli w tworzeniu skrótów

Sól jest dodatkowym ciągiem danych dodawanym do tekstu przed wyznaczeniem skrótu, co powoduje, że nawet 2 identyczne hasła generują inne skróty. Sól zwiększa zabezpiecza przed atakami typu „rainbow table” korzystających z bazy skrótów, która pozwala na zaoszczędzenie mocy obliczeniowej koniecznej do złamania hasła metodą „brute force” W mojej implementacji nie została ona uwzględniona, co mogłoby prowadzić do potencjalnych luk w bezpieczeństwie.

### 4. Czy funkcję MD5 można uznać za bezpieczną

Funkcja MD5 jest uznawana za przestarzałą i niebezpieczną ze względu na znalezione kolizje oraz wykazaną podatność na ataki. Nie zaleca się stosowania MD5 w sytuacjach, w których bezpieczeństwo danych jest istotne.

## 5. Zestawienie wyników



### Wnioski:

- Czas działania MD5 jest dłuższy od pozostałych funkcji, co najlepiej widać przy ciągach dłuższych ciągach wejściowych
- Generowanie skrótów dla długich ciągów zajmuje więcej czasu niż dla krótkich ciągów.
- Prawdopodobieństwo wystąpienia kolizji na pierwszych 12 bitach skrótu wyznaczonego przy użyciu SHA-256 wynosi ok. 3,6%
- Losowość wyjścia funkcji skrótu SHA-256 wynosi 48,44%, czyli jest zbliżona do idealnego wyniku, którym byłoby 50%