

Funkcje skrótu

Sebastian Nowak - 152065

1. Screenshot wyników

```
Text: h
SHA3-512: 0.000024s, 26cfb46b8264aa515069b0726c0ed4d1c08587a2f1572fcee6a06b6611ba7802e657791c8e64bf372042bd86208995a9a2a1ef2248d1202137ae65b0906f1ae3
SHA-256 : 0.000011s, aaa9402664f1a41f40ebbc52c9993eb66aeb366602958fdfaa283b71e64db123
SHA-1   : 0.000029s, 27d5482eebd075de44389774fce28c69f45c8a75
MD5     : 0.000004s, 2510c39011c5be704182423e3a695e91

Text: String
SHA3-512: 0.000003s, b6f8e8f6b90446f9d7d73a54ef8fb71f5f0ed496338c25f09cf6aae5823f611c52c360242f987f9bf99b4d4d696c651fe261aec760b1e5b7cc2ac2bfaa51f982
SHA-256 : 0.000024s, b2ef230e7f4f315a28cdcc863028da31f7110f3209feb76e76fed0f37b3d8580
SHA-1   : 0.000001s, 3df63b7acb0522da685dad5fe84b81fdd7b25264
MD5     : 0.000001s, 27118326006d3829667a400ad23d5d98

Text: Ala ma psa
SHA3-512: 0.000003s, e0d8650d39bb456f2777d1dd04f62158e5ce48868cbcb0b6e96a849718d4e3c2ed0df6f999fa242a31f68355094eca2565e44d3feff5e37f6c68a7c675ae30d5
SHA-256 : 0.000002s, d39eac5aff09d4b8077a6c824d70394539b1542af9c6b88eac99a87d52859496
SHA-1   : 0.000001s, a9c7a2a4e21952f10e70858950270709fb12837b
MD5     : 0.000001s, 11fd81f319abd1bac49d596d61cfe769

Text: Nazywam sie Jan
SHA3-512: 0.000002s, cda1b16692ead79a45f2b74654f0d979dead53a0c6315763368139c3669940f11cb53833f37ae1db034b14bd62cdcfe0c30ff2550332023b9b4f9553bb78a627
SHA-256 : 0.000002s, ea9774c699bc0424a3758d701c350cd3c2708567fabd2492dd842fdb50602af8
SHA-1   : 0.000001s, d8d843c5492b3aa237162d6c3eac014ed238692f
MD5     : 0.000001s, 9bff548c39d73526f37de5780a5a16d9

Text: To jest testowy tekst i jest dlugi
SHA3-512: 0.000020s, 98ad007ece86e56ac5d9974a72914ae110fae4cfc3482f139926bdfa1ef2fdefb0e4320474b0a53b807ab98c8e9c8f37326a901c009afeabd39f1939d1fae09
SHA-256 : 0.000002s, 6d810bcacebd6d92589d4ed8b25cbf570ce0c92d4c20983751d0cd97f19b948d6
SHA-1   : 0.000001s, 16a48d056f9745d3e28a156eadcd40fae42cd6d6
MD5     : 0.000001s, f8752d35e238b16d2dd3572c31ef74a0

MD5 hash of 'teemo' is 85ce0d6a319c09b564e2bd32cc64e38f
Probability of collision in first 12 bits: 0.041
Tested SAC score: 0.5156
```

2. Sposób implementacji

Aplikacja została napisana w pythonie. Do wszystkich funkcji skrótu została wykorzystana biblioteka *hashlib*. Na screenie powyżej widać porównanie czasów różnych funkcji skrótu oraz badanie kolizji i losowości hash'u funkcji SHA-256.

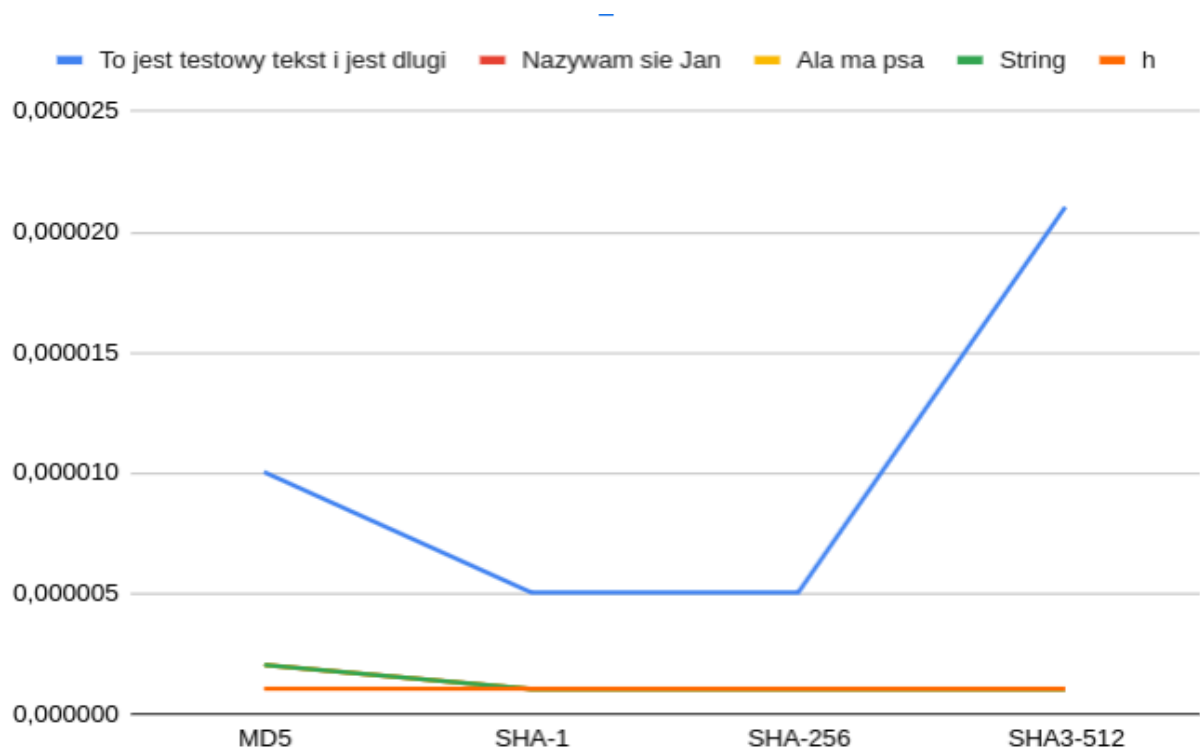
3. Rola soli w powstawaniu skrótów

Sól, dodatkowy ciąg danych dodawany do tekstu przed wyznaczeniem skrótu, powoduje, że nawet dwa identyczne hasła generują różne skróty. Sól zwiększa zabezpieczenia przed atakami typu „rainbow table”, które korzystają z bazy skrótów, co pozwala zaoszczędzić moc obliczeniową potrzebną do złamania hasła metodą „brute force”. W mojej implementacji nie uwzględniono soli, co mogłoby prowadzić do potencjalnych luk w bezpieczeństwie.

4. Bezpieczność funkcji MD5

Znalezione kolizje oraz wykazana podatność na ataki sprawiają, że funkcja MD5 jest uznawana za przestarzałą i niebezpieczną. Nie zaleca się stosowania MD5 w sytuacjach, w których bezpieczeństwo danych jest kluczowe.

5. Wnioski



- MD5 dłużej się wykonuje niż pozostałe funkcje, co jest szczególnie widoczne przy dłuższych ciągach wejściowych.
- Generowanie skrótów dla długich ciągów zajmuje więcej czasu niż dla krótkich.
- Prawdopodobieństwo wystąpienia kolizji na pierwszych 12 bitach skrótu wyznaczonego przy użyciu SHA-256 wynosi około 4,1%.
- Losowość wyjścia funkcji skrótu SHA-256 wynosi 51,56%, co oznacza, że jest zbliżona do 50%.