Algorytmy Hashowania

Kamil Filipiński, 66140

1 Omówienie problemu

Algorytmy hashowania są fundamentalnymi narzędziami w kryptografii i innych dziedzinach informatyki. Umożliwiają one efektywne przechowywanie, przetwarzanie i zabezpieczanie danych. Hashowanie jest procesem przekształcania danych wejściowych (np. hasła) na ciąg znaków o ustalonej długości, zwany hashem. Hashe są wykorzystywane m.in. do przechowywania haseł, sprawdzania integralności danych oraz w kryptowalutach. W tym kontekście kluczowe jest zapewnienie, aby hash był odporny na ataki kryptograficzne, takie jak ataki brute-force i kolizje.

2 Przegląd algorytmów hashowania

2.1 B-Crypt

Zastosowania: B-Crypt jest najczęściej używany do bezpiecznego przechowywania haseł. Jego odporność na ataki brute-force i rainbow table sprawia, że jest preferowany w aplikacjach wymagających wysokiego poziomu bezpieczeństwa.

Działanie:

- B-Crypt bazuje na algorytmie Blowfish i jest specjalnie zaprojektowany do ochrony haseł.
- Wykorzystuje salt - losowe dane dodawane do hasła przed jego hashowaniem, co sprawia, że dwa identyczne hasła będą miały różne hashe.
- Zawiera parametr kosztu, który kontroluje, jak długo trwa proces hashowania. Wyższy koszt oznacza dłuższy czas hashowania, co zwiększa bezpieczeństwo przed atakami brute-force.

B-Crypt jest szczególnie ceniony za swoją elastyczność i możliwość dostosowania parametrów hashowania, co pozwala na zwiększenie bezpieczeństwa wraz ze wzrostem mocy obliczeniowej atakujących.

Implementacja:

```
import bcrypt

# Hashowanie hasla
```

```
password = b"supersecret"
hashed = bcrypt.hashpw(password, bcrypt.gensalt())

# Weryfikacja hasla
if bcrypt.checkpw(password, hashed):
    print("Password matches")
else:
    print("Password does not match")
```

2.2 SHA-256

Zastosowania: SHA-256 jest szeroko używany w kryptowalutach (np. Bitcoin), cyfrowych podpisach, certyfikatach SSL oraz w innych zastosowaniach wymagających bezpiecznego hashowania danych.

Działanie:

- SHA-256 należy do rodziny SHA-2 i jest zaprojektowany przez NSA.
- Produkuje 256-bitowe (32-bajtowe) hashe, co zapewnia wysoki poziom bezpieczeństwa.
- Algorytm jest odporny na ataki kolizyjne, co oznacza, że znalezienie dwóch różnych wejść dających ten sam hash jest bardzo trudne.

SHA-256 jest powszechnie używany ze względu na swoje bezpieczeństwo i wydajność. Jest kluczowym elementem w technologii blockchain, gdzie zapewnia integralność transakcji.

Implementacja:

```
import hashlib

# Hashowanie wiadomosci
message = "hello world"
hashed = hashlib.sha256(message.encode()).hexdigest()

print(hashed)
```

2.3 MD5

Zastosowania: MD5 był kiedyś powszechnie używany do sprawdzania integralności plików i danych oraz przechowywania haseł, ale ze względu na swoje słabe strony, jest teraz uważany za przestarzały.

Działanie:

- Produkuje 128-bitowe (16-bajtowe) hashe.
- Znany jest z podatności na kolizje, co oznacza, że jest możliwe znalezienie dwóch różnych wejść dających ten sam hash.

- Został zaprojektowany przez Ronalda Rivesta w 1991 roku jako następca algorytmu MD4.

MD5 jest obecnie uważany za niewystarczająco bezpieczny do większości zastosowań kryptograficznych. Pomimo to, jest nadal używany w niektórych starszych systemach ze względu na swoją prostotę i szybkość.

Implementacja:

```
import hashlib

# Hashowanie wiadomosci
message = "hello world"
hashed = hashlib.md5(message.encode()).hexdigest()

print(hashed)
```

2.4 SHA-1

Zastosowania: SHA-1 był kiedyś szeroko używany do cyfrowych podpisów i certyfikatów SSL, ale z powodu znanych słabości bezpieczeństwa, jego użycie jest obecnie odradzane.

Działanie:

- Produkuje 160-bitowe (20-bajtowe) hashe.
- Znany jest z podatności na kolizje, a ataki praktyczne na SHA-1 zostały zaprezentowane, co znacznie zmniejszyło zaufanie do jego bezpieczeństwa.
- SHA-1 został zaprojektowany przez NSA i opublikowany w 1993 roku.

Pomimo znanych problemów bezpieczeństwa, SHA-1 jest nadal używany w niektórych starszych systemach i aplikacjach, które nie zostały jeszcze zaktualizowane.

Implementacja:

```
import hashlib

# Hashowanie wiadomosci
message = "hello world"
hashed = hashlib.sha1(message.encode()).hexdigest()

print(hashed)
```

2.5 Blake2

Zastosowania: Blake2 to nowoczesny algorytm hashowania używany do szerokiego zakresu zastosowań, takich jak podpisy cyfrowe, generowanie kluczy i inne zastosowania kryptograficzne.

Działanie:

- Blake2 jest szybszy niż MD5, SHA-1 i SHA-256, a jednocześnie zapewnia wysoki poziom bezpieczeństwa.
- Dwa warianty: Blake2b (dla 64-bitowych systemów) i Blake2s (dla 32-bitowych systemów).
- Blake2 został zaprojektowany, aby być prostym w implementacji, a jednocześnie bardzo bezpiecznym i wydajnym.

Blake2 jest szeroko stosowany w nowoczesnych systemach kryptograficznych ze względu na swoją efektywność i bezpieczeństwo.

Implementacja:

```
import hashlib

# Hashowanie wiadomosci
message = "hello world"
hashed = hashlib.blake2b(message.encode()).hexdigest()

print(hashed)
```

3 Czas łamania algorytmów

Poniższa tabela przedstawia szacowany czas łamania różnych algorytmów hashowania dla wyrażeń o różnej długości:

Algorytm	Czas łamania (8-znakowe haslo)	Czas łamania (16-znakowe haslo)
B-Crypt	Kilka lat	Miliony lat
SHA-256	Kilka godzin	Kilka tysięcy lat
MD5	Minuty	Kilka lat
SHA-1	Godziny	Kilka tysięcy lat
Blake2	Kilka godzin	Kilka tysięcy lat

Warto zauważyć, że czas łamania hasła zależy od wielu czynników, takich jak moc obliczeniowa atakującego, długość i złożoność hasła oraz dodatkowe zabezpieczenia, takie jak użycie soli.

4 Wnioski

Algorytmy hashowania różnią się pod względem bezpieczeństwa i wydajności. B-Crypt jest obecnie uważany za jeden z najbezpieczniejszych algorytmów do przechowywania haseł, jednak jego użycie wiąże się z wyższym kosztem obliczeniowym. SHA-256, choć bardzo bezpieczny, jest mniej odporny na ataki bruteforce w porównaniu do B-Crypt. Algorytmy takie jak MD5 i SHA-1, kiedyś popularne, są teraz uważane za przestarzałe i niebezpieczne z powodu znanych podatności na kolizje. Blake2 oferuje nowoczesne podejście, łącząc wysoką wydajność z bezpieczeństwem.

5 Referencje

- https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes
- https://www.dobreprogramy.pl/programy-dla-studentow-jak-rozpoczac-prace-z-latex-em,6628431494375041a
- https://maumneto.medium.com/git-vs-code-overleaf-91ecfd586b36
- https://en.wikipedia.org/wiki/Cryptographic_hash_function
- https://en.wikipedia.org/wiki/Bcrypt
- https://en.wikipedia.org/wiki/SHA-2
- https://en.wikipedia.org/wiki/MD5
- https://en.wikipedia.org/wiki/SHA-1
- https://en.wikipedia.org/wiki/BLAKE_(hash_function)