Algorytmy hashowania – mechanizmy i zastosowania

Olgierd Gerszyński, indeks 73715

30 kwietnia 2024

Wstep

Algorytmy hashowania pełnia kluczowa role w ochronie danych w erze cyfrowej, przekształcajac dowolne dane wejściowe w unikalny, stałodługościowy ciag bitów zwanym skrótem, który jest praktycznie niemożliwy do odwrócenia. Te algorytmy zapewniaja zarówno szybkość przetwarzania danych, jak i ich bezpieczeństwo, co czyni je niezastapionymi narzedziami w systemach zabezpieczeń, bazach danych oraz w aplikacjach internetowych.

Problem i zastosowanie algorytmów hashowania

Algorytmy hashowania rozwiazuja kilka fundamentalnych problemów w dziedzinie bezpieczeństwa danych i szybkiego dostepu do informacji: bezpieczne przechowywanie haseł, weryfikacja integralności danych oraz optymalizacja wyszukiwania.

Omówienie pieciu algorytmów hashowania

MD5

MD5, choć niegdyś szeroko stosowany, jest obecnie uważany za przestarzały z powodu poważnych słabości, które umożliwiaja generowanie kolizji w rozsadnie krótkim czasie. Generuje on 128-bitowy skrót.

SHA-1

Podobnie jak MD5, SHA-1 generuje 160-bitowy skrót i jest obecnie uważany za niewystarczajaco bezpieczny ze wzgledu na możliwość szybkiego generowania kolizji.

SHA-256

Należacy do rodziny SHA-2, SHA-256 jest szeroko stosowany w bezpiecznych systemach, generujac 256-bitowy skrót. Jest znacznie bardziej odporny na ataki niż jego poprzednicy.

B-Crypt

B-Crypt jest specjalnie zaprojektowany do bezpiecznego hashowania haseł. Jego unikalna cecha to mechanizm solenia i regulacja czasu obliczeń, co czyni go odpornym na ataki siłowe nawet przy wykorzystaniu nowoczesnego sprzetu do przetwarzania.

SHA-3

Najnowsza iteracja w rodzinie Secure Hash Algorithms, SHA-3 oferuje jeszcze wieksza bezpieczeństwo niż SHA-2 i jest zalecana do wszystkich nowych zastosowań wymagajacych funkcji skrótu.

Przykładowe implementacje

Przykład implementacji algorytmu SHA-256 w jezyku Python:

```
import hashlib

def hash_input(input_string):
    return hashlib.sha256(input_string.encode()).hexdigest()

example_string = "Testowy string"
hashed_string = hash_input(example_string)
print("SHA-256:", hashed_string)
```

Kalkulacja złożoności obliczeniowej

Algorytmy hashowania sa zaprojektowane tak, aby były szybkie w generowaniu skrótu, ale jednocześnie wolne i kosztowne obliczeniowo w przypadku prób odwrócenia.

Wnioski

Algorytmy hashowania stanowia fundament bezpieczeństwa cyfrowego, umożliwiajac ochrone danych przed nieautoryzowanym dostepem oraz ich szybka weryfikacje.

Referencje

- Metody ataków oraz podstawowe algorytmy szyfrowania w cyberbezpieczeństwie
- Funkcja skrótu – Wikipedia
- What is Hashing? – Binance Academy
- Rodzaje algorytmów szyfrujacych oraz ich implementacje programowe
- Szyfry blokowe - PDF
- Podrecznik algorytmy - PDF