Funkcje skrótu

Katarzyna Jaromirska 155910

# Screenshot z aplikacji:

Poniższy zrzut ekranu przedstawia wyniki procesu haszowania wiadomości z pliku. Dla każdej linii uruchamiane są wszystkie funkcje skrótu.



# Omówienie sposobu implementacji

Wykorzystana została biblioteka java.security do haszowania wiadomości. Program bierze bajty z wiadomości, a następnie przepuszcza przez instancję danej funkcji skrótu. Wiadomość może być pobrana z konsoli lub z pliku ”hash\_input.txt”. W konsoli wyświetlana jest zhaszowana wiadomość w dostępnych funkcjach, tj.: MD5, SHA-1, rodzina SHA-2 oraz rodzina SHA3.

W celu wykonania pomiarów użyto dodatkowo biblioteki benchmarkowej org.openjdk.jmh. Biblioteka ta pozwala na profesjonalny pomiar czasu poszczególnych funkcji.

## Użycie funkcji skrótu

Klasa Hash.java zawiera kolejno: nazwy algorytmów, funkcję wypisywania wyników, funkcję haszującą za pomocą wybranego algorytmu oraz dodatkowa konwersja bajtów na tekst

public class Hash {  
 public static final String[] ALGORITHMS = {  
 "MD5", "SHA-1",  
 "SHA-224", "SHA-256", "SHA-384", "SHA-512", "SHA-512/224", "SHA-512/256",  
 "SHA3-224", "SHA3-256", "SHA3-384", "SHA3-512" };  
  
 public static void output(String input) {  
 for(String algorithm : ALGORITHMS) {  
 try {  
 String hash = bytesToHex(hashMessage(input, algorithm));  
 System.out.printf("%-13s: %s%n", algorithm, hash);  
 } catch (NoSuchAlgorithmException e) {  
 System.out.printf("%-10s: No support.%n", algorithm);  
 System.out.println("Error: " + e.getMessage());  
 }  
 }  
 }  
  
 public static byte[] hashMessage(String input, String algorithm)

throws NoSuchAlgorithmException {  
 byte[] messageBytes = input.getBytes(StandardCharsets.UTF\_8);  
 MessageDigest md = MessageDigest.getInstance(algorithm);  
  
 return md.digest(messageBytes);  
 }  
  
 private static String bytesToHex(byte[] bytes) {  
 StringBuilder hexString = new StringBuilder();  
  
 for (byte b : bytes) {  
 String hex = Integer.toHexString(0xff & b);  
  
 if (hex.length() == 1)  
 hexString.append('0');  
  
 hexString.append(hex);  
 }  
  
 return hexString.toString();  
 }  
}

# Rola soli w tworzeniu skrótów

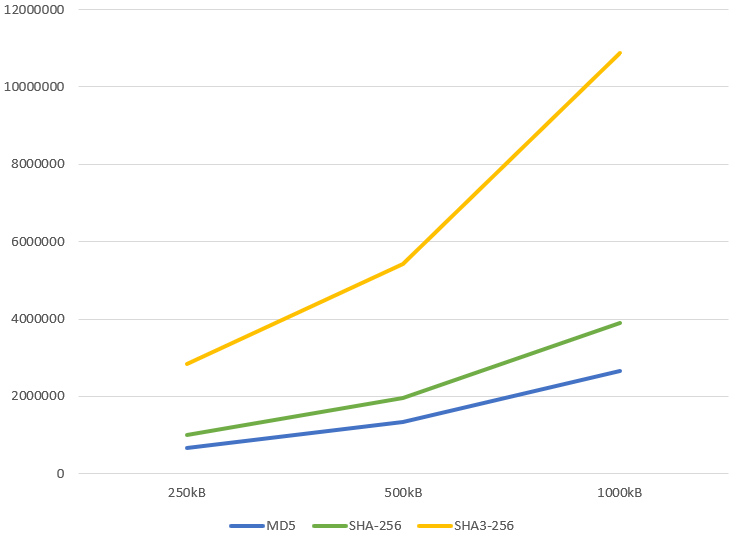
Sól w kryptografii to losowe dane dodawane do hasła. Hashowanie z użyciem soli pomaga w zapobieganiu przed atakami z użyciem tablic tęczowych (baza skrótów używana do łamania haszowanych haseł). Sprawia również, że dwa takie same hasła będą miały różne wyjście po haszowaniu.

# Czy funkcja MD5 jest bezpieczna? Czy znaleziono kolizje?

Funkcja MD5 nie jest bezpieczna. Jest uznawana za kryptograficznie złamaną funkcję haszującą. Kolizje zostały już znalezione w 2004 przez chińskich naukowców. Oznacza to, że istnieją dwie różne wiadomości M i M’ dla których h(M) = h(M’).

# Wyniki pomiaru czasu funkcji skrótu

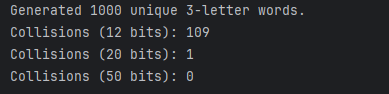
Zmierzono czasy wybranych trzech funkcji skrótu: MD5, SHA-256 oraz SHA3-256. Poniżej znajdują się kolejno wykresy pomiaru czasu funkcji w zależności od rozmiaru wiadomości. Biblioteka benchmarkowa języka Javy oferuje również wielkość błędu pomiarów, jednakże wartości nie były warte dodania do poniższej analizy.



Jak można było przeczuwać, algorytm z rodziny SHA3 (obecnie najbezpieczniejszy) jest najwolniejszy. Czas wykonywania rośnie niemal kwadratowo. Natomiast dla MD5 i SHA-256 czasy są podobne, małe. Można zauważyć, że im większe rozmiary tym większe różnice pomiędzy czasami algorytmów.

# Badanie kolizji na bitach

Sprawdzono kolizję na pierwszych 12 bitach, 20 bitach oraz 50 bitach dla 1000 różnych 3-literowych słów. Użyty został algorytm haszujący SHA-256.



## Sprawdzenie kryterium SAC (Strict AValanche Criteria)

