***Wojskowa Akademia Techniczna***

***im. Jarosława Dąbrowskiego***

Laboratorium z przedmiotu:

[Wprowadzenie](http://shaql.w.staszic.waw.pl/~shaql/wcy/viewforum.php?f=13) do Kryptologii

Sprawozdanie z ćwiczenia laboratoryjnego nr 2:

**Funkcje skrótu**

Prowadzący:

mgr inż. Marta Turowska

**Wykonał:** Radosław Relidzyński

**Grupa:** WCY20IY4S1

**Data laboratoriów**: 29.04.2021 r.

Spis treści

[A. Treść zadania 2](#_Toc102316175)

[B. Testowanie wektorów 2](#_Toc102316176)

[C. Generowanie skrótu dla pliku pdf 3](#_Toc102316177)

[D. Generowanie skrótu dla zmienionego pliku pdf 4](#_Toc102316178)

[E. Porównanie i wnioski 5](#_Toc102316179)

# Treść zadania

1. Przetestować wektory pod kątem zwracania dobrego wyniku.
2. Wygenerować skrót dla pliku z prezentacją.
3. Zmienić 1 bajt w pliku.
4. Wygenerować skrót dla nowego pliku.
5. Wyciągnąć wnioski.

# Testowanie wektorów

Korzystam ze zbioru wyników z repozytorium pod poniższym linkiem:

<https://github.com/XKCP/XKCP/blob/master/tests/TestVectors/ShortMsgKAT_SHA3-224.txt>

Do testowania będę używał polecenia:

*printf "$message" | openssl dgst -sha3-224 -binary | xxd -p | tr a-f A-F*

Lista kroków przy obliczaniu funkcji skrótu:

1. W miejsce *$message* należy wpisać interesującą nas wiadomość.
2. *printf* konwertuje wiadomość na kod ASCII.
3. *openssl dgst -sha3-224 -binary* oblicza skrót przy użyciu algorytmu sha3-224 i zwraca go w zapisie binarnym.
4. *xxd -p* konweruje do postaci hexadecymalnej.
5. *tr a-f A-F* zmienia litery na wielkie , żeby otrzymany wynik porównać z tym w pliku z repozytorium.

Tesowane wartości (wraz z długością L):

1. Brak wiadomości (L = 0)
2. CC (L = 8)
3. 41FB (L = 16)
4. 1F877C (L = 24)
5. C1ECFDFC (L = 32)

Zwrócone wartości:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Sprawdzenie poprawności wyniku:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Jak widać dla każdego wyniku zwracana wartość jest zgodna z oczekiwaną

# Generowanie skrótu dla pliku pdf

Korzystam z okna openssl.exe

W ramach niego wywołam opcję:

*sha3-224 $file*

gdzie w miejsce *$file* wpiszę konkretny plik wraz ze ścieżką dostępu do niego

Zwrócona wartość:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Przy pomocy zwykłej konsoli zamieniam litery na wielkie:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Otrzymany skrót:

**01B2AC2433AFF53AFC462E95520E45A55D78DEA89C44C5605F9D89AE**

# Generowanie skrótu dla zmienionego pliku pdf

Przy pomocy rozszerzenia „hex editor” otwieram skopiowany plik pdf i zmieniam w nim pierwszy bajt

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Generuję skrót dla nowego pliku pdf:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Widać już, że znacząco różnią się od siebie

Przy pomocy zwykłej konsoli zamieniam litery na wielkie:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Otrzymany skrót:

**DBC945E298DC7DF33BE810C34A45EDE70870E7AC5103BDCABE681E10**

# Porównanie i wnioski

Dla pewności zrobię porównanie dwóch skrótów i sprawdzę, czy są jakiekolwiek podobieństwa

Obraz zawierający tekst, zrzut ekranu, ekran

Opis wygenerowany automatycznie

Tylko na 2 pozycjach są takie same znaki, co wskazuje na to, że są to 2 zupełnie różne od siebie ciągi znaków.

Jak więc widać, nawet 1 bajt różnicy powoduje, że openssl generuje w pełni różne wyniki.

Dobra funkcja haszująca posiada na tyle skomplikowany model kodowania informacji, żeby nie dało się w żaden sposób ocenić, jakie mechanizmy ma w sobie zastosowane.

Załóżmy scenariusz, że człowiek chcący odkryć model kodowania ma zbiór zwróconych wartości na podstawie zadanych tekstów. Na przykładzie różnic między skrótami pdf-ów można stwierdzić, że niezależnie od tego, czy teksty będą różniły się całym słowem, czy tylko literą, otrzymane wartości będą drastycznie inne i bez możliwości porównania ich. Uniemożliwia to znajdowania zależności do określania modelu kodowania.

Oczywiście nie zawsze model jest niewykrywalny. Źle opracowana funkcja haszująca może nie zapewniać bezpieczeństwa danych i może być możliwa do odkrycia, a czasami nawet możliwa do zastosowania wstecz (do dekodowania informacji).