|  |  |
| --- | --- |
| **Politechnika Białostocka**  **Wydział Informatyki** | Data: 12.10.2016 |
| **Przedmiot:** Techniki zapewniania poufności w Internecie.  **Sprawozdanie nr:** 1  **Temat:** JEDNOKIERUNKOWE FUNKCJE HASHUJĄCE. ALGORYTM SHA-1.  **Autor:** Maciej Ziniewicz  **Studia:** stacjonarne II stopnia, semestr 2 | **Prowadzący:**  prof. dr hab. Vyacheslav Yarmolik  Ocena: |

Spis treści

[1. Treść zadania 2](#_Toc464143197)

[2. Część teoretyczna 2](#_Toc464143198)

[3. Rozwiązanie 3](#_Toc464143199)

# Treść zadania

Wykonaj oprogramowanie realizujące hashowanie algorytmem SHA-1 wybranego pliku.

# Część teoretyczna

SHA-1 (Secure Hash Algorithm) jest jednokierunkową funkcją hashującą wytwarzającą skrót o długości 160 bitów z wiadomości o dowolnym rozmiarze, nie większym niż -bitów. Funkcja SHA-1 jest oparta na zasadach podobnych, do użytych przy projektowaniu funkcji hashujących MD4 i MD5, i w znacznym stopniu naśladuje te algorytmy.

Początkowo wiadomość jest uzupełniana o podobnie jak w MD5 czyli dołączany jest bit o wartości 1 na końcu wiadomości, następnie ciąg uzupełnia się w taki sposób by składał się z 512-bitowych bloków, a ostatni blok był niepełny i miał dokładnie 448 bitów. Na koniec ostatniego bloku dodawana jest 64-bitowa wartość oznaczająca rozmiar wiadomości przed modyfikacjami. Po dodaniu rozmiaru ostatni blok równiez będzie miał 512 bitów. Następnie wykonywana jest głowna pętla algorytmu tyle razy ile jest 512 bitowych bloków.

Każdy blok jest 512 bitowy i taki pojedynczy blok jest przetwarzany w głownej pętli. 512 bit daje 16 – 32 bitowych wartości(słowa od M0 do M15) , które są przekształcane w 80 – 32 bitowych wartości (od W0 do W79) w taki sposób że pierwsze 16 jest przepisywanych tak jak we wzorze :

Z kolei pozostałe od indeksu 16 do 79 są określone wzorem:

Kolejnym etapem jest przepisane stałych które należało wcześniej zdefiniować:

A = 0x67452301

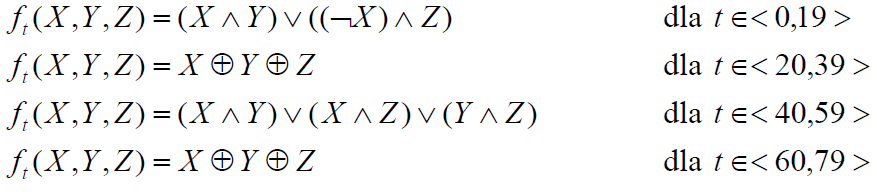
B = 0xefcdab89

C = 0x98badcfe

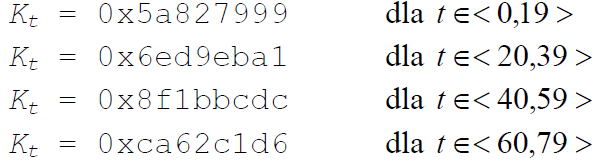
D = 0x10325476

E = 0xc3d2e1f0

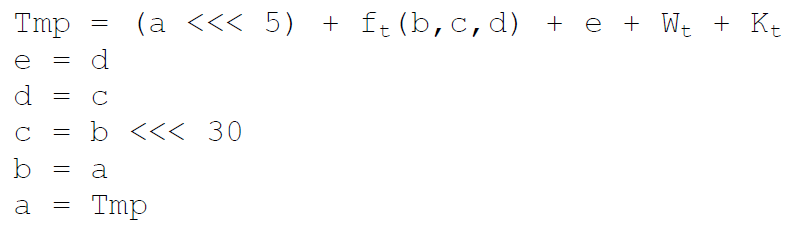
Przepisać przykładowo na a,b,c,d,e. Na tych przepisanych stałych będą wykonywane operacje pętli. Kolejnym etapem jest wykonanie nieliniowych funkcji, wykonywane są na określony zakres indeksów z wcześniej utworzonych osiemdzieśięciu 32 bitowych wartości.



W algorytmie zdefiniowane są też stałe, również przyporządkowane do indeksów.



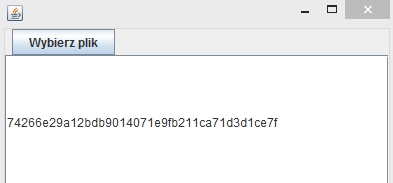
Mając te zakresy można przystąpic do głownej pętli algorytmu która wykonuje się dla każdego 32bitowego podbloku o indeksie 0-79. Wedle zakresów podanych powyżej oraz dla każdego przejścia wykonywane są dodatkowo następujące operacje



Po wykonaniu wszystkich działań zmienne a, b, c, d i e są dodawane odpowiednio do zmiennych A, B, C, D i E, a algorytm jest kontynuowany dla następnego bloku wiadomości. Ostatecznie wyjściem algorytmu jest konkatenacja zmiennych A, B, C, D i E.

# Rozwiązanie

Stworzona oprogramowanie zostało napisane w języku Java i działa w bardzo prosty sposób. Po uruchomieniu otwierane jest okienko gdzie mamy możliwość wybrania pliku którego zawartość chcemy zaszyfrować, po wybraniu pliku zatwierdzamy go i jest on automatyczie szyfrowany, a wynik szyfrowania jest podawany w polu tekstowym.

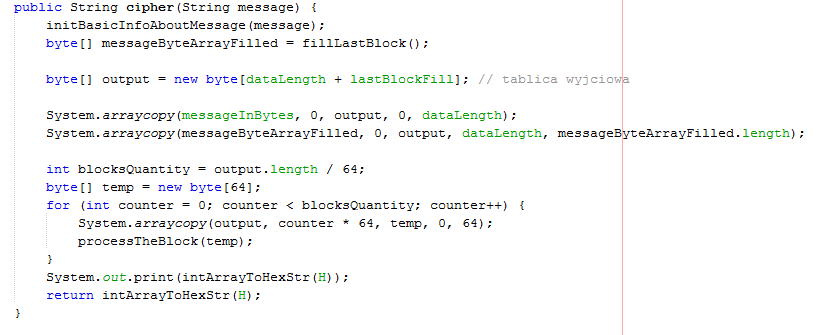


W programie została napisana klasa SHA1 która odpowiada za cały procesz szyfrowania.

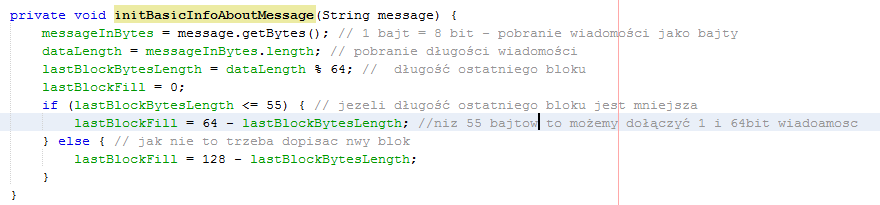


Powyższe zmienne to stałe zadeklarowane, wymagane do działania algorytmu.

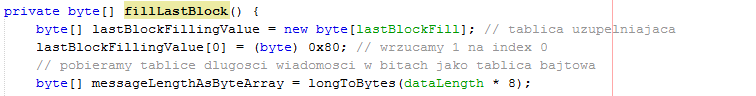
Do zaszyfrowania wiadomości należy użyć metody cipher(parametr) gdzie jako parametr należy podać wiadomość jako tekst. Metoda ta zwróci zaszyfrowaną wiadomość również w formie tekstowej.



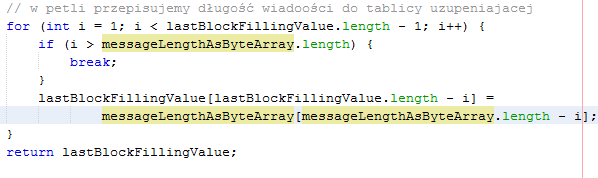
Powyższa metoda rozpoczyna swoje działanie od ustawienia podstawowych informacji o podanej wiadomości używając funkcji „initBasicInfoAboutMessage(parametr)” gdzie jako parametr podawana jest ta wiadomość.



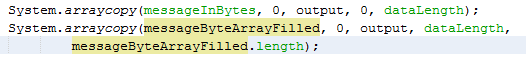
Funkcja „initBasicInfoAboutMessage” działa tak jak to opisują komentarze na powyższym zrzucie ekranu. Pobiera wiadomość jako tablice bajtową, określa długośc wiadomości, określa długość ostatniego bloku oraz jego wypełnienie.

Kolejnym etapem szyfrowania oraz kolejną funkcją w programie jest wypełnienie ostatniego bloku za pomocą funkcji „fillLastBlock()”.

Pierwsza częsc funkcji tworzy tablicę o wielkośći długości wypełnienia, na pierwszy jej indeks wrzuca wartość 1, jest to tablica bajtowa dlatego należy dodać wartość 0x80 co w bitach jesy wyrażone jako 1000000. Kolejna linijka wykonuje metodę „longToBytes” która zamienia watrość liczbową na bity, wartość jest podawana pomnożona razy 8 ponieważ, zapisana długość to długość w bajtach a do szyfrowania potrzeba jest długość w bitach.

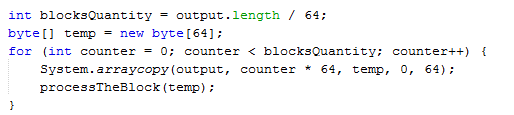


Następnie wykonywana jest pętla która przepisze w odpowiednie miejsca od wartości tablicy z informacją o długości wiadomości tablicy do tablicy utworzonej wcześniej z dodaną już wartością 1 na początku, w ten sposób otrzymaliśmy wypełnienie wiadomości.

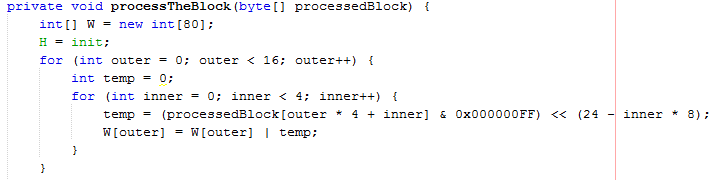


Kolejnymi krokami w programie jest skopiowanie tablicy z wiadomością do tablicy wyjsciowej, a następnie dodanie utworzonej wczesniej tablicy z wypełnieniem do tablicy wyjściowej. Otrzymana została w ten sposób wiadomość wraz z wypełnieniem czyli 1 i 64bit informacją o długości wiadomości na końcu.

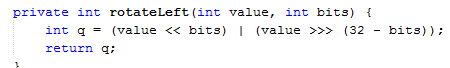
Kolejny etap to wykonanie głownej pętli funkcji SHA-1. Początkowo dzielę cała tablicę wyjsciową na 64 bajty czyli 512 bit i otrzymuje ilość bloków.



Następnie wykonuję pętlę tyle razy ile bloków ma wiadomość. W pętli kopiowane jest odpowiednie 64 bajtów wiadomości do tablicy pomocniczej. Po czym wywołuję funkcję przetwarzająca „processTheBlock(parametr)” gdzie jako parametr podawany jest 512 bitowy blok wiadomości.

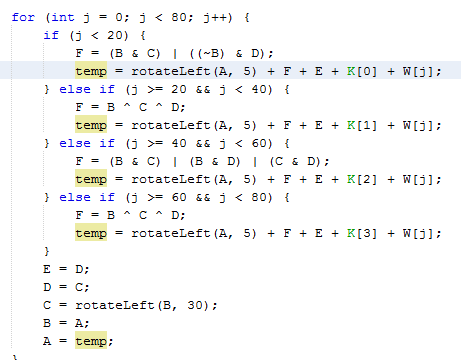


Funkcja początkowo z tablicy bajtów(8bit element) tworzy 16 elementową tablicę intów(32bit element) a następnie rozszerza ją do 80 elemetnowej tablicy za oraz przesuwa w lewo z pomocą funkcji rotującej „rotateLeft”. 

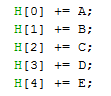


Funkcja ta tworzy wartość przesuniętą w lewo o podaną ilość bitów oraz drugą wartość przesuniętą w prawo o podana ilość bitów i wypełnioną zerami. Wykonywana jest operacja OR na tych wartościach, dzięki czemu uzyskujemy przesuniętą w lewo o 1 bit podaną wartość. 

Po przesunięciu w lewo przepisywane są wartości A,B,C,D,E na których będziemy wykonywać działania. A następnie wykonywana jest pętla z funkcjami oraz operacjami szyfrującymi.



Dla każdego z zakresów tablicy tj [0,19),(20,39),(40,59),(60,79)] wykonywana jest odpowiednia funkcja wedle wzorów podanych w części teoretycznej, a następnie dla każdego już zakresu wykonywane są przestawienia również podane w części teoretycznej.



Po wykonanu wszystkich działań funkcje A,B,C,D,E są przepisywane do zmiennych z których były kopiowane i proces przetwarzający wykonywany jest dla kolejnego bloku.

Ostatecznie wyjściem algorytmu są zmienne zawarte w tablicy H. Które po zamianie na formę tekstową są wyświetlane w okienku.

Poprawność zaimplementowania algorytmu sprawdzana z generatorem internetowym.

