|  |  |
| --- | --- |
| **#**  implementacja funkcji skrótu sha-3  Projekt z przedmiotu Zaawansowane metody kryptografii i ochrony informacji [MKOI] | Temat projektu  Implementacja funkcji skrótu SHA-3 (wszystkie warianty) oraz porównanie jej wydajności z istniejącą implementacją  Wykonujący projekt  inż. Paweł Kamiński  inż. Jan Muczyński  PROWADZĄCY PROJEKT  mgr inż. Albert Sitek  Semestr: 17Z |

1. Wprowadzenie

Tematem projektu jest implementacja funkcji skrótu SHA-3. Jest to najnowsza funkcja w rodzinie standardów *Secure Hash Algorithm*. Została wyłoniona w wyniku konkursu przeprowadzonego przez NIST w 2015 roku [1]. Algorytmem, który został wybrany był algorytm *Keccak.*

Funkcje skrótu są powszechnie używane na świecie w wielu dziedzinach informatyki. Służą przekształceniu wiadomości cyfrowej na krótki „skrót” wiadomości. Wykorzystuje się je między innymi przy tworzeniu podpisów cyfrowych. Za ich pomocą można również np. porównać dwa pliki i jeśli oba zwracają taką samą wartość skrótu to z bardzo dużym prawdopodobieństwem można stwierdzić, że są one takie same i nie zostały zmodyfikowane (integralność). Wykorzystywane są również często do uwierzytelnienia wiadomości. Innym przykładem zastosowania takich funkcji jest to, że w bazach danych systemów informatycznych zazwyczaj zapamiętywane są skróty z haseł użytkowników, zamiast konkretnych ciągów znaków. W przypadku wykradzenia takiej bazy danych, atakujący nadal nie ma możliwości odczytać danych służących do uwierzytelnienia przez użytkownika.

Dobre funkcje skrótu powinny posiadać trzy cechy:

* Odporność na kolizje – brak możliwości wygenerowania dwóch dowolnych wiadomości o tym samym skrócie
* Odporność na kolizje konkretnych wiadomości – brak możliwości wygenerowania wiadomości o takim samym skrócie jak wskazana wiadomość
* Jednokierunkowość – brak możliwości wnioskowania o wiadomości na podstawie jej skrótu. Zmiana pojedynczego bita w wiadomości powinna znacząco zmieniać wartość skrótu.

SHA-3 powstało w 4 wariantach, każdy różniący się długością skrótu. Są to długości: 224 bity, 256 bitów, 384 bity, 512 bitów.

1. Narzędzia i struktura implementacji

Implementacja funkcji skrótu SHA-3 została zrealizowana w języku C++ jako biblioteka współdzielona DLL. Podczas tworzenia oprogramowania zostało użyte środowisko programistyczne Visual  Studio 2017. W ramach projektu została również napisana aplikacja konsolowa w języku C++, umożliwiająca wyliczanie skrótów SHA-3.

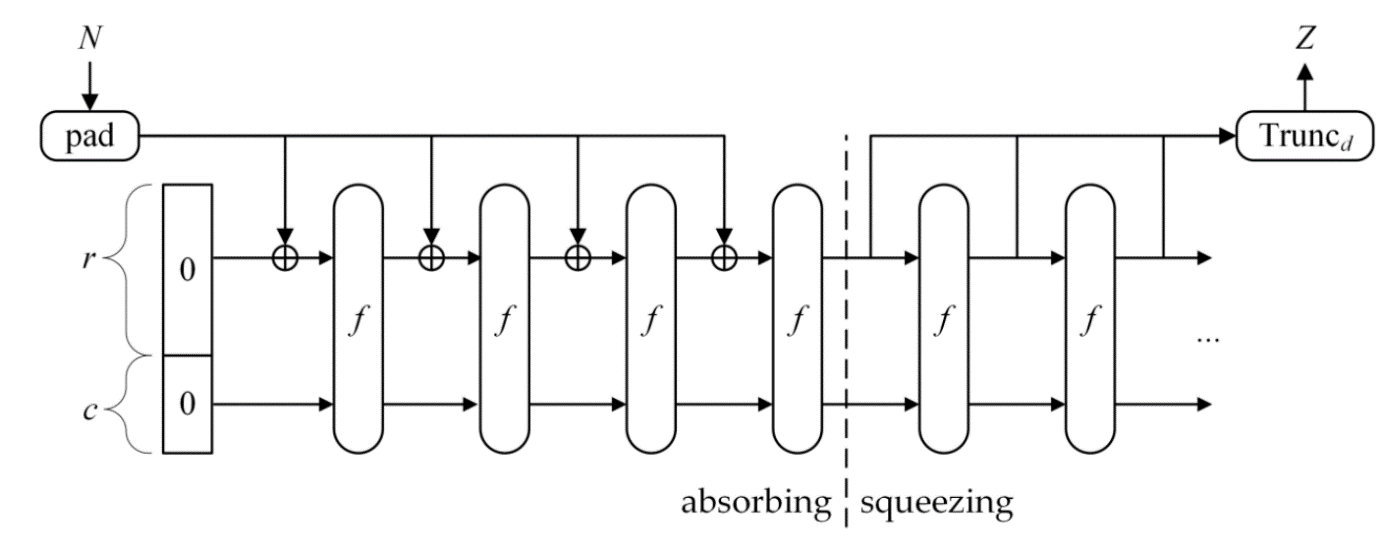
Aplikacja okienkowa, która wywołuje funkcje SHA-3 z biblioteki współdzielonej, została napisana w języku Python. Na okienku użytkownik ma opcję wyboru pliku, rodzaju algorytmu oraz przycisk uruchamiający procedurę obliczania skrótu SHA-3.

Biblioteka współdzielona generuje plik z logami, do którego zapisywane są informacje o przebiegu działania algorytmu.

Do przetestowania wydajności działania SHA-3 został napisany program w języku Python, który oblicza funkcje skrótu z wielu plików. Korzysta w tym celu z zaimplementowanej przez nas biblioteki oraz z biblioteki libcrypto++. Program oblicza wszystkie warianty długościowe SHA-3 oraz generuje wykresy porównujące wydajność obu bibliotek.

1. Opis algorytmu

Szczegółowy opis algorytmu SHA-3 został zamieszczony w dokumencie FIPS-202 [2]. Nazwa algorytmu implementującego SHA-3 to *Keccak*. Ma on strukturę gąbki (*sponge)*. Analogia do gąbki polega na tym, że bity wejściowe są „wchłaniane” do stanu funkcji, po czym pewna ilość bitów wyjściowych jest „wyciskana” ze stanu funkcji.



Rysunek 1 Konstrukcja gąbki

Funkcja sponge przyjmuje ciąg bitów *N* oraz długość wyjściowego ciągu *d* (w bitach). Ciąg *N* jest uzupełniany o padding. Funkcja *f* to funkcja mieszająca stanu. Składa się z 24 rund, podczas których wykonywane jest 5 funkcji:

* Θ (theta) – miesza odpowiednie bity ze sobą
* ρ (rho) – stosuje rotacje bitowe
* π (Pi) – przestawienie bitów
* χ (Chi) – mieszanie bitów
* ι (Jota) – przekształcenie niektórych bitów leżących w Lane[0,0] o wartość zależącą od rundy

Ciało funkcji *f* wygląda w pseudokodzie następująco:

keccakPermutation(String m, StateArray A)

convertStringToStateArray(m, A);

for (i = 0; i < 24; i++)

keccakTheta(A);

keccakRho(A);

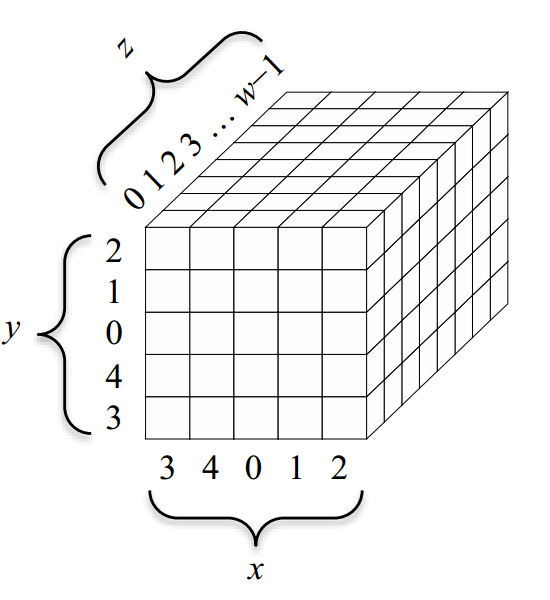
keccakPi(A);

keccakChi(A);

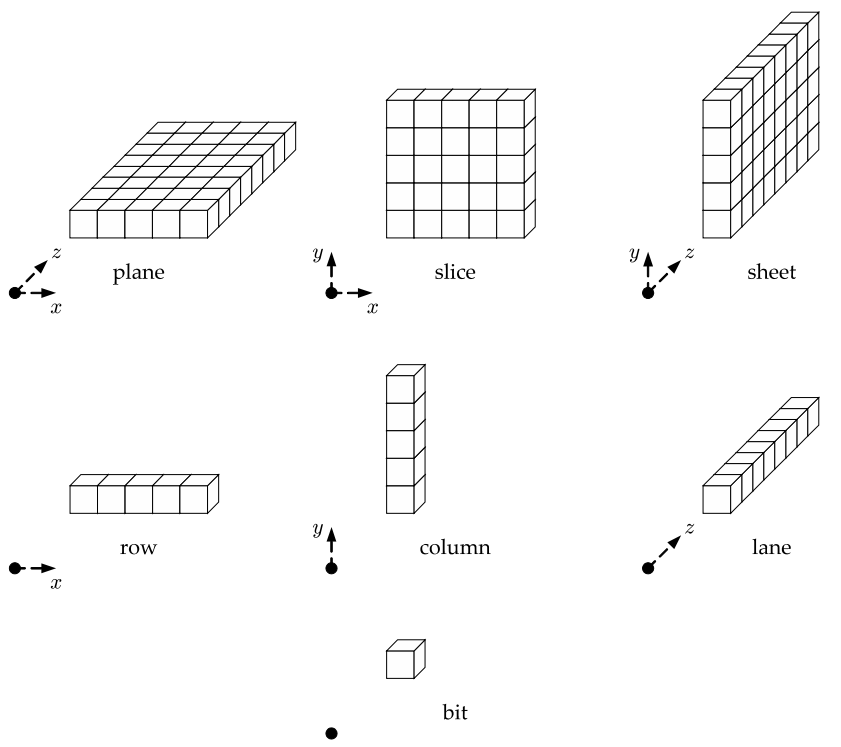
keccakJota(A, round);

return convertStateArrayToString(A);

Tablica stanu:

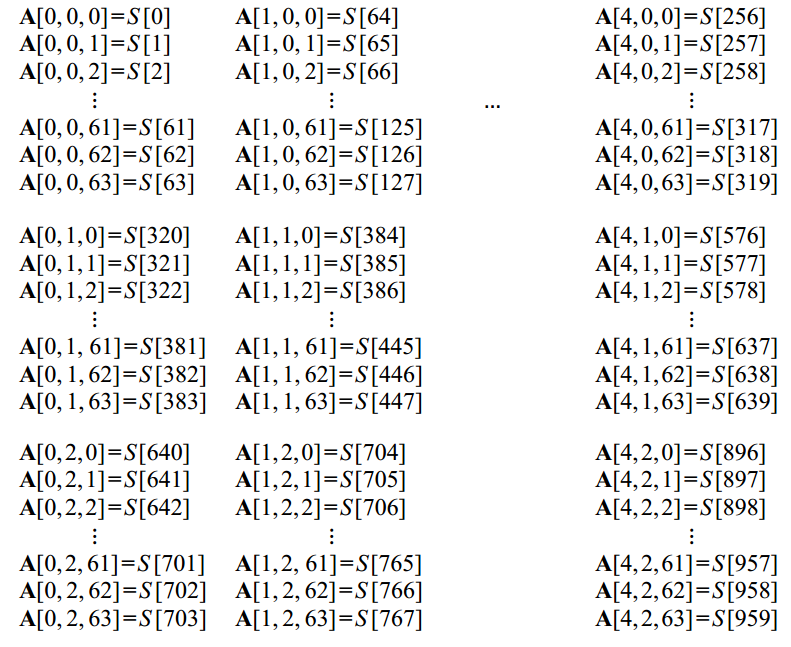


Rysunek 2 Tablica stanu wraz z opisem współrzędnych



Rysunek 3 Części tablicy stanu pod względem wymiaru

Sposób tworzenia tablicy stanu A z ciągu bitów S został przedstawiony poniżej:



# Bibliografia

|  |  |
| --- | --- |
| [1] | NIST, „NIST,” 2015. [Online]. Available: https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard. |
| [2] | NIST, „FIPS PUB 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions,” Gaithersburg, MD 20899-8900, 2015. |