Programowanie równoległe i rozproszone

Algorytm szyfrujący AES w trybie CTR

Maciej Stefańczyk, Kacper Szkudlarek

6 grudnia 2010

Streszczenie

Dokumentacja realizacji projektu z przedmiotu "Programowanie równoległe i rozproszone". W ramach projektu wykonana została implementacja algorytmu szyfrującego AES (Advanced Encryption Standard) pracującego w trybie CTR (Counter mode). Zrównoleglenie zostało oparte o bibliotekę OpenMP dla pamięci wspólnej, oraz MPI dla pamięci rozproszonej.

1 Wstęp

Algorytm szyfrujący AES (ang. Advanced Encryption Standard) jest to symetryczny szyfr blokowy przyjęty przez NIST (ang. National Institute of Standards and Technology) jako standard szyfrowania. Algorytm został stworzony w ramach ogłoszonego konkursu, który miał wyłonić następcę algorytmu DES(ang. Data Encryption Standard). Jego robocza nazwa to Rijndael. Ze względu na swą złożoność algorytm uważany jest za bezpieczny i powszechnie stosowany.

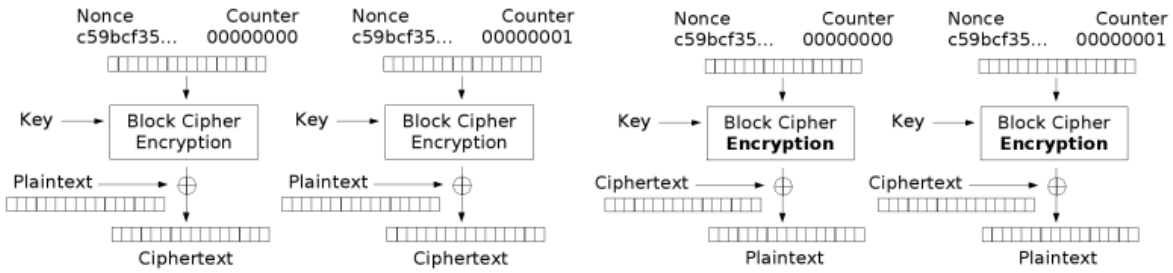
Algorytm szyfruje i deszyfruje dane w 128-bitowych blokach, korzystając z 128, 192 lub 256 bitowych kluczy. Dane, na których operuje algorytm formowane są w macierz 4x4, w której każda komórka jest jednym bajtem danych. W zależności do wybranej wielkości klucza szyfrującego wykonywane jest odpowiednio 10, 12 lub 14 rund szyfrujących. Wszystkie rundy poza pierwszą i ostatnią składają się z czterech kroków:

1. Etap: substytucja wstępna - jest to nielinowe przekształcenie każdego bajtu poprzez zamianę jego wartości zgodnie z tablicą LUT.

2. Etap: zamiana wierszy - na każdym wierszu danych wykonywana jest operacja rotacji cyklicznej o zadaną liczbę pozycji.

3. Etap: mieszanie kolumn - każda kolumna poddawana jest odwracalnej liniowej transformacji.

4. Etap: dodanie klucza rundowego - do danych dodawany jest klucz rundowy wygenerowany na podstawie klucza głównego.



Rysunek 1: Schemat działania trybu licznikowego.

W opisywanej implementacji algorytm szyfrowania AES pracował w trybie CTR (ang . Counter). Jest to tzw. tryb licznikowy. Pozwala wykorzystać on szyfry blokowe do kodowania m.in. strumieni danych. Generowany jest pseudolosowy ciąg danych, który który pełni rolę strumienia szyfrującego. Jest on mieszany poprzez użycie operacji XOR z danymi wejściowymi, przez co uzyskujemy zaszyfrowany ciąg danych. Zostało to przedstawione na rysunku (1).

2 Implementacja

W ramach projektu powstały trzy wersje oprogramowania opisane w dalszych paragrafach:

1. Wersja sekwencyjna - jest to implementacja algorytmu w języku C wykonująca się jednowątkowo w sposób sekwencyjny.

2. Wersja wykorzystująca pamięć współdzieloną - jest to rozwinięcie wersji sekwencyjnej, które dzięki wykorzystaniu OpenMP pozwala na równoległe wykonywanie na wielu procesorach.

3. Wersja wykorzystująca pamięć rozproszoną - jest to rozwinięcie wersji sekwencyjnej, które dzięki wykorzystaniu protokołu MPI pozwala na równoległe wykonywanie na wielu maszynach połączonych między sobą siecią.

2.1 Wersja sekwencyjna

Diagram przepływu danych dla stworzonej aplikacji został przedstawiony na rysunku (2). Na podstawie danych podanych w wierszu poleceń w czasie wykonywania programu podejmowane są decyzje o trybie działania - szyfrowanie, deszyfrowanie, długości wykorzystanego klucza, pobierane są dane wejściowe.

Dane podawane do programu (klucz, dane do szyfrowania/deszyfrowania) odczytywane są bezpośrednio z plików i wczytywane w całości do pamięci. Dzięki takiemu rozwiązaniu narzut czasowy związany z odczytem i zapisem danych w czasie ich szyfrowania jest stosunkowo niewielki, gdyż wszystkie operacje przeprowadzane są na pamięci, a nie na plikach dyskowych. Powoduje to jednak wydłużenie czasu wczytywania danych, zwłaszcza przy dużych zbiorach, które maja być zaszyfrowane.

Pseudolosowa liczba wykorzystywana w trybie CTR do szyfrowania strumienia danych została zaimplementowana w postaci złożenia czterech 32 bitowych bloków. Najbardziej znaczące 32 bit wypełniane jest wartością czasu odczytaną w momencie ładowania do pamięci pliku z danymi do zaszyfrowania. Kolejne dwa bloki 32 bitowe stanowią liczby odpowiednio 1 i 0. Do najmniej znaczących 32 bit wpisywana jest wartość licznika zliczającego ilość zaszyfrowanych bloków. W czasie zapisywania wyników wartość górnych 64 bitów zapisywana jest na samym początku pliku wynikowego.

Wykorzystane w środkowej części liczy pseudolosowej wartości 1 i 0 w celu zwiększenia bezpieczeństwa należałoby zastąpić jakimiś bardziej skomplikowanymi wartościami.

W celu optymalizacji przetwarzania danych zostały wprowadzone pewne modyfikacje w sposobie przechowywania i zapisywania macierzy stanu algorytmu. Każda kolumna przechowywana jest w pamięci w postaci 32 bitowej liczby. Takie potraktowanie kolumn pozwoliło znacząco ułatwić proces "mieszania" kolumn w trakcie szyfrowania.

2.2 Pamięć wspólna – OpenMP

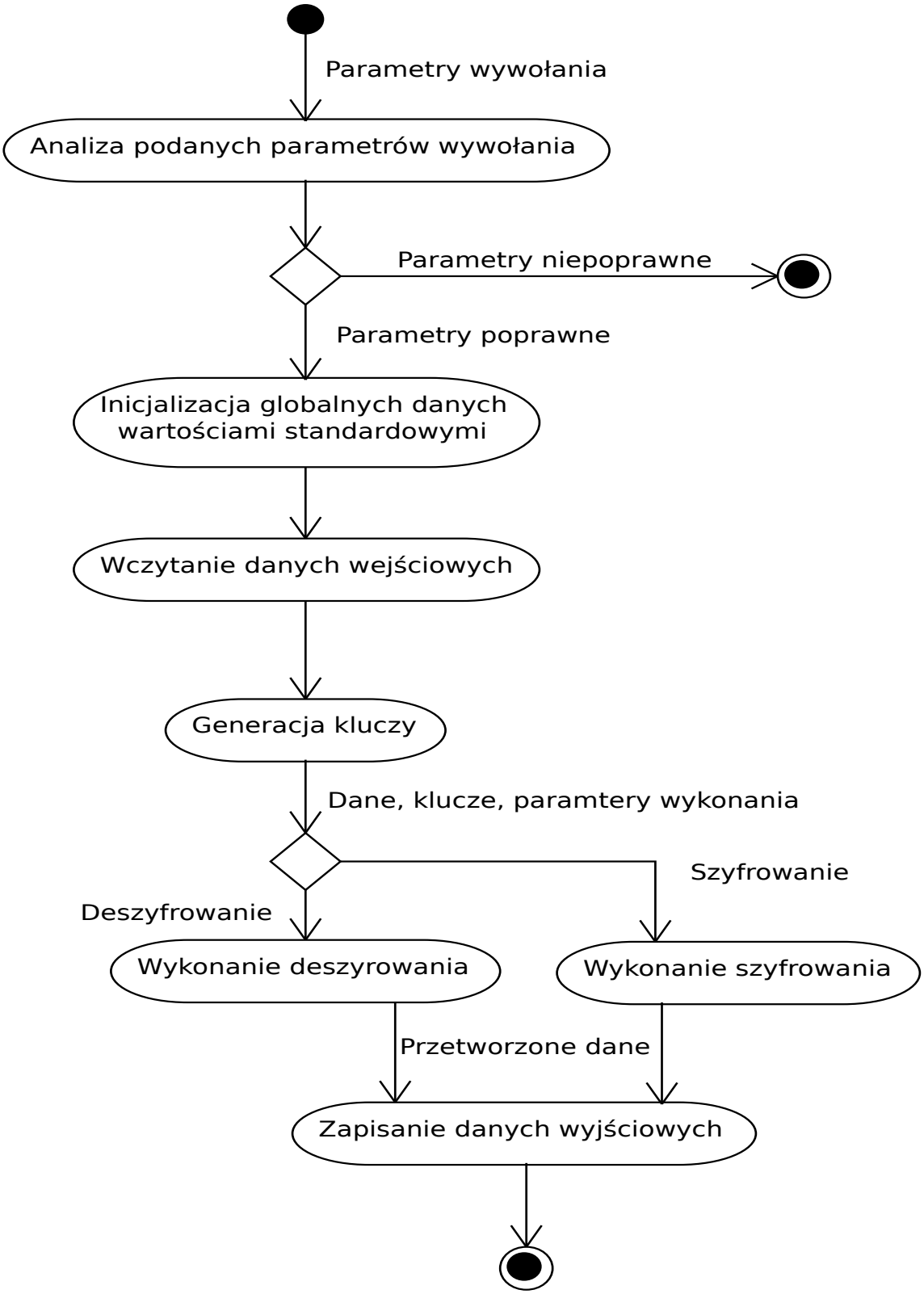
OpenMP (ang. Open Multi-Processing) jest to wieloplatformowy interfejs programowania aplikacji umożliwiający tworzenie aplikacji dla systemów wieloprocesorowych korzystających z pamięci współdzielonej. Interfejs może być wykorzystywany w językach C, C++ i FORTRAN, na architekturach Unix i Windows. Do sterowania sposobem wykonywania programu używa się zbiory dyrektyw kompilatora oraz zmiennych środowiskowych.

Dzięki zastosowaniu trybu licznikowego CTR w implemnetacji algorytmu AES możliwe było bardzo dobre zrównoleglenie przeprowadzanych obliczeń. W tym trybie każdy 128 bitowy blok danych szyfrowany jest niezależnie od pozostałych. Jedynym elementem, który należy kontrolować jest takie modyfikowanie licznika, by możliwe było późniejsze odtworzenie jego pracy w celu odszyfrowania danych.

W celu oznaczenia bloku do zrównoważenia użyta została następująca dyrektywa:
`#pragma omp parallel for default(none) private(i) shared(data, in_blocks, in_ptr32)`
Odnosząca się do następującego bolku kodu:

```
1     for (i = 0; i < in_blocks; ++i)
2     {
3         aes_state_t ctr = aesCipherCounter(data, i);
4         uint32_t c0 = *(uint32_t*)ctr.s;
5         uint32_t c1 = *(uint32_t*)ctr.s+4;
6         uint32_t c2 = *(uint32_t*)ctr.s+8;
7         uint32_t c3 = *(uint32_t*)ctr.s+12;
8         in_ptr32[4 * i    ] = in_ptr32[4 * i    ] ^ c0;
9         in_ptr32[4 * i + 1] = in_ptr32[4 * i + 1] ^ c1;
10        in_ptr32[4 * i + 2] = in_ptr32[4 * i + 2] ^ c2;
11        in_ptr32[4 * i + 3] = in_ptr32[4 * i + 3] ^ c3;
12    }
```

Dyrektywa ta definiuje prywatne i współdzielone zmienne zawarte w wyróżnionym fragmencie. Jako prywatna została oznaczona zmienna sterująca pętla `for`, tak by każdy wątek mógł niezależnie przetwarzać dane. Zmiennymi publicznymi są natomiast:



Rysunek 2: Diagram przepływu danych w czasie wywołanie programu.

- `data` - struktura danych zawierająca zmienne globalne,
- `in_blocks` - liczba bloków, które należy przetworzyć,
- `un_ptr32` - wskaźnik na przetwarzane dane.

Zrównoleglona pętla `for` odpowiada za szyfrowanie algorytmem AES kolejnych chwilowych zmiennych losowych i przeprowadzanie operacji XOR tych zmiennych z szyfrowanymi danymi, a także operację odwrotną - deszyfrowanie.

2.3 Pamięć rozproszona – MPI

MPI (ang. Message Passing Interface) jest to protokół komunikacyjny będący standardem przesyłania komunikatów pomiędzy procesami programów równoległych korzystających z pamięci rozproszonej - działających na jednym lub więcej komputerach połączonych w sieć.

TODO: słoniu opisz to!!

2.4 Dodatkowe możliwości

2.5 Obsługa programu

3 Testy

3.1 Testy jednostkowe

3.2 Testy wydajności

3.2.1 Dane testowe

3.2.2 Wyniki

Tabela 1: Czas przetwarzania pojedynczej ramki obrazu

	1280x720		960x540		640x360		320x180	
	czas [ms]	FPS	czas [ms]	FPS	czas [ms]	FPS	czas [ms]	FPS
FraDIA	**45,7**	21,9	**26,8**	37,3	**12,7**	78,7	**4,5**	222,2
RAW	**44,5**	22,5	**26,1**	38,3	**12,1**	82,6	**4,3**	232,6