T^3 - Twórca Tablic Tęczowych

Równoległe wyznaczanie tęczowych tablic ("rainbow tables") w zgadnieniach kryptografii dla haseł zaszyfrowanych algorytmem DES: Scala

Bartosz Pieńkowski, Barnaba Turek

30 maja 2011

1 Opis

T^3 to zestaw programów wyznaczających tablice tęczowe i pozwalających sprawdzić poprawność ich wyznaczenia (przez wyznaczenie funkcji skrótu dla danego ciągu znaków (dalej klucza) i próbę odwrócenia tego procesu).

Tablice tęczowe to sposób przechowywania wcześniej obliczonych danych pozwalających odwracać (analitycznie nieodwracalną) funkcję skrótu[1]. *Tablice tęczowe* pozwalają zmniejszyć wymagania dyskowe (w stosunku do prostego zapisywania wszystkich par klucz-f(klucz)) kosztem wymagań obliczeniowych. Osiągane to jest za pomocą tworzenia tzw. łańcuchów skrótów[2] i zapisywaniu tylko pierwszego i ostatniego elementu łańcucha.

1.1 Funkcja skrótu

T^3 wyznacza *tablice tęczowe* dla funkcji skrótu **md5**.

2 Użycie

2.1 Tworzenie tablic tęczowych

Program generujący tablice tęczowe nazywa się *t3*.

2.1.1 Wywołanie programu

Użytkownik programu t3 podaje trzy argumenty linii poleceń. Pierwszy argument określa długość klucza, dla którego mają być wygenerowane *tablice tęczowe* (od 1 do 8). Drugi argument określa długość obliczanych łańcuchów. Trzeci argument określa plik, w którym mają być zapisane klucze.

[1] inaczej kryptograficzną funkcję mieszającą, ang. *hash function*
[2] ang. *hash chaining*

2.1.2 Wyjście programu

Tablice tęczowe zostaną zapisane w podanym pliku. Program tworzy plik, składający się z wierszy. Każdy wiersz składa się z początkowego i końcowego elementu łańcucha skrótów, oddzielnoych spacją. Łańcuchy są uporządkowane wg. końcowego łańcucha, aby przyspieszyć wyszukiwanie. Na początku pliku znajduje się nagłówek określający długość łańcuchów, alfabet z którego są zbudowane i długość kluczy.

2.1.3 Przykładowe wywołania

Wywołanie:

```
1  $ t3 2 10
```

Wygeneruje tablice tęczowe dla haseł o dłguości dwóch znaków. Łańcuchy skrótów będą miały długość 10.

Do generowania skrótów służy program *t3-hash*.

2.1.4 Wywołanie programu

Użytkownik programu *t3-hash* jako argument podaje klucz (1 do 8 znaków), dla którego ma być wygenerowany skrót. Następnie program wypisuje skrót na standardowe wyjście.

2.2 Odwracanie funkcji skrótu

Do odwracania funkcji skrótu służy program *t3-reverse*.

2.2.1 Wywołanie programu

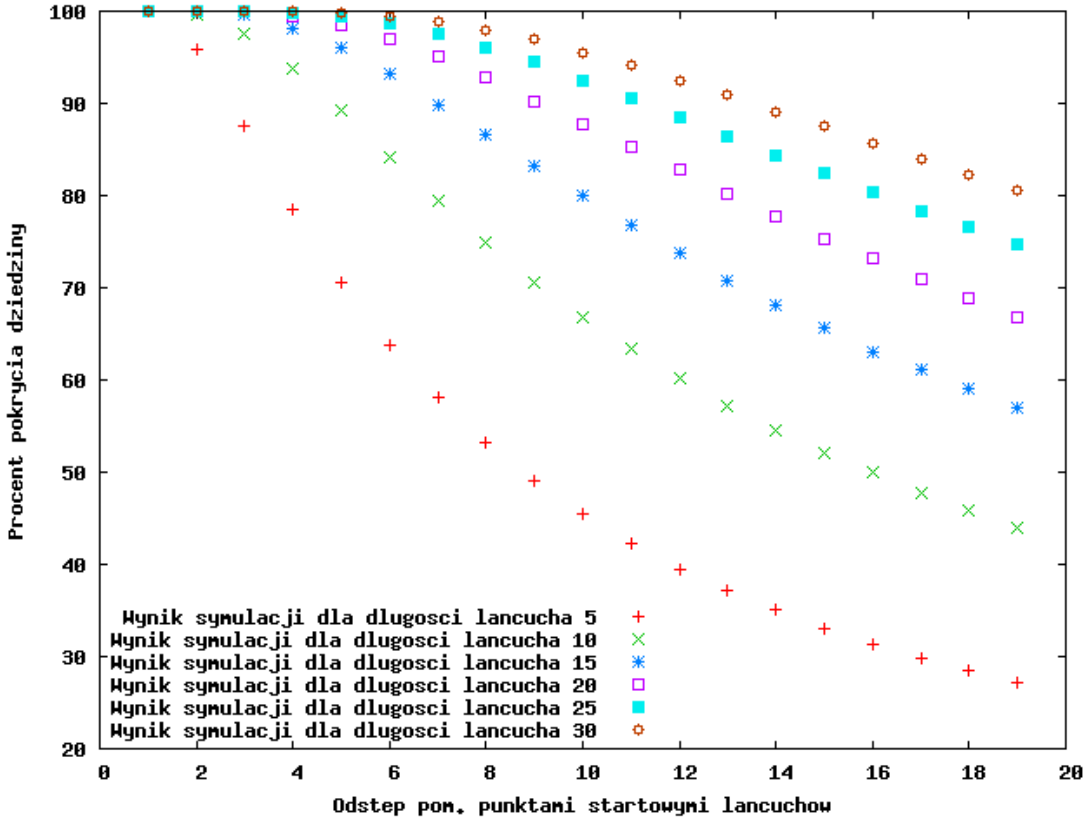
Użytkownik programu *t3-reverse* jako argument podaje wartość funkcji skrótu, dla której znaleziona ma być wartość klucza.

Jako drugi argument należy podać ścieżkę do pliku z tablicami tęczowymi wygenerowanymi przez T^3 .

Jeżeli program nie znajdzie klucza pasującego do zadanej wartości funkcji skrótu, program zakończy działanie wypisując informację o niepowodzeniu.

Jeżeli działanie programu zakończy się sukcesem, program wypisze znaleziony klucz. Poprawność znalezionego klucza będzie można sprawdzić korzystając z programu *t3-hash*.

2.2.2 pokrycie zbioru kluczy



3 Implementacja

3.1 Rozwiązania

3.1.1 Redukcja

Wynik funkcji MD5 to 16 bajtów. Możemy to zapisać jako 4 zmienne typu int.

```
1     for (int i=0; i<4; i++){
        hash[i] = getInt(MD5Bytes.slice(i*4,(i+1)*));
3       hash[i] = idx ^ (redux_number << (i*4));
      }
```

Następnie z każdej zmiennej tworzymy cztery indeksy do tablicy reprezentującej alfabet.

AlphaLen-liczba elementów w tablicy.

```
val pwd = Buffer[Int];
for (int i=0; i<4; i++){
   pwd += (hash[i] % AlphaLen^2) / AlphaLen;
   pwd += (hash[i] % AlphaLen^2) % AlphaLen;
   pwd += ((hash[i] / AlphaLen^2) % AlphaLen^2) / AlphaLen;
   pwd += ((hash[i] / AlphaLen^2) % AlphaLen^2) % AlphaLen;
}

//mieszanie w~celu zroznicowania kolejnych redukcji

if (redux_number > 16)
  pwd = pwd.reverse();

val slice = pwd.slice(0,redux_number%16);
pwd.trimStart(redux_number);

if (redux_number > 32)
  pwd = pwd.reverse();
elseif (redux_number > 48)
  slice = slice.reverse();

pwd += slice;
```

Następnie elementy pwd są mapowane na znaki ze słownika i żądana liczba elementów z początku bufora pwd jest zwracana jako nowy klucz.

Tak zaplanowana funkcja redukcji powinna działać dośc dobrze dla alfabetów nie dłuższych niż 84 znaki.

3.1.2 Generowanie tablic

Zrównoleglenie wyznaczania tablic tęczowych zostanie osiągnięte za pomocą mechanizmu Aktorów oferowanego przez język **Scala**. Mechanizm ten jest zrealizowany na wirtualnej maszynie Javy za pomocą wątków.

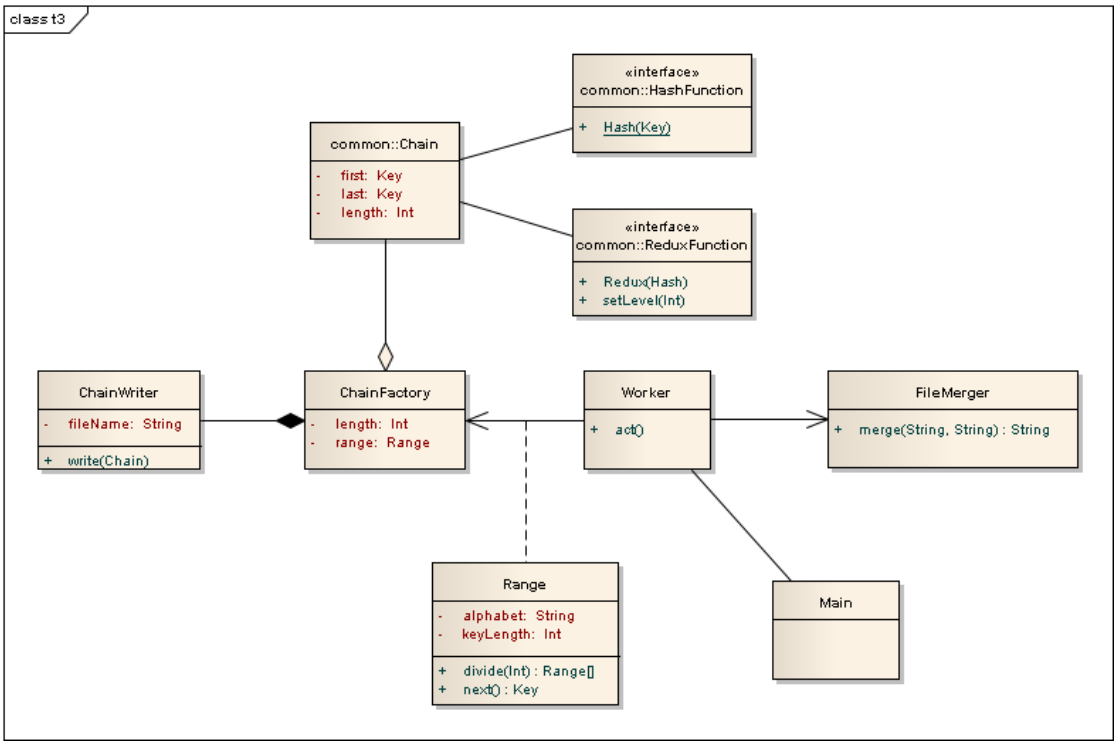
Zamierzamy wykorzystać komunikację globalną - jeden wątek będzie zarządzał wszystkimi innymi wątkami.

Naszym zdaniem najlepszą dekompozycją problemu przy tak postawionym zadaniu będzie dekompozycja domenowa, tj. równomierny podział początkowych[3] kluczy na wątki.

Klucze będą dzielone na paczki o określonym, stałym rozmiarze. Na początku aktor zarządzający procesem wyśle wiadomość do każdego aktora-wykonawcy. Wiadomość będzie zawierała informacje o początkowych kluczach, dla których mają zostać wygenerowane łańcuchy. Wykonawcy po obliczeniu wszystkich kluczy, których obliczenie zostało im zlecone, wyślą do zarządcy wiadomość zawierającą nazwę pliku tymczasowego, w którym zapisali posortowane łańcuchy. Zarządca doda nazwę pliku do kolejki (FIFO) plików, które muszą zostać połączone.

Następnie, kiedy cały zadany zakres kluczy zostanie już znaleziony, zostaną utworzeni nowi aktorzy, odpowiedzialni za scalanie plików. Zarządca wyśle do każdego z nich

[3] zaczynających łańcuch skrótów



Rysunek 1: diagram klas programu T^3

wiadomość zawierającą nazwy dwóch pierwszych plików z kolejki, które należy scalić. Ponieważ dane w plikach są posortowane, scalanie będzie się odbywać na zasadzie podobnej do scalania w algorytmie merge-sort. Po skończeniu scalania dwóch plików, aktor zapisze scalone dane do trzeciego pliku, usunie dwa pliki wejściowe i prześle zarządcy wiadomość zawierającą nazwę nowego pliku. Zarządca odbierze wiadomość i doda plik do kolejki plików wymagających scalenia.

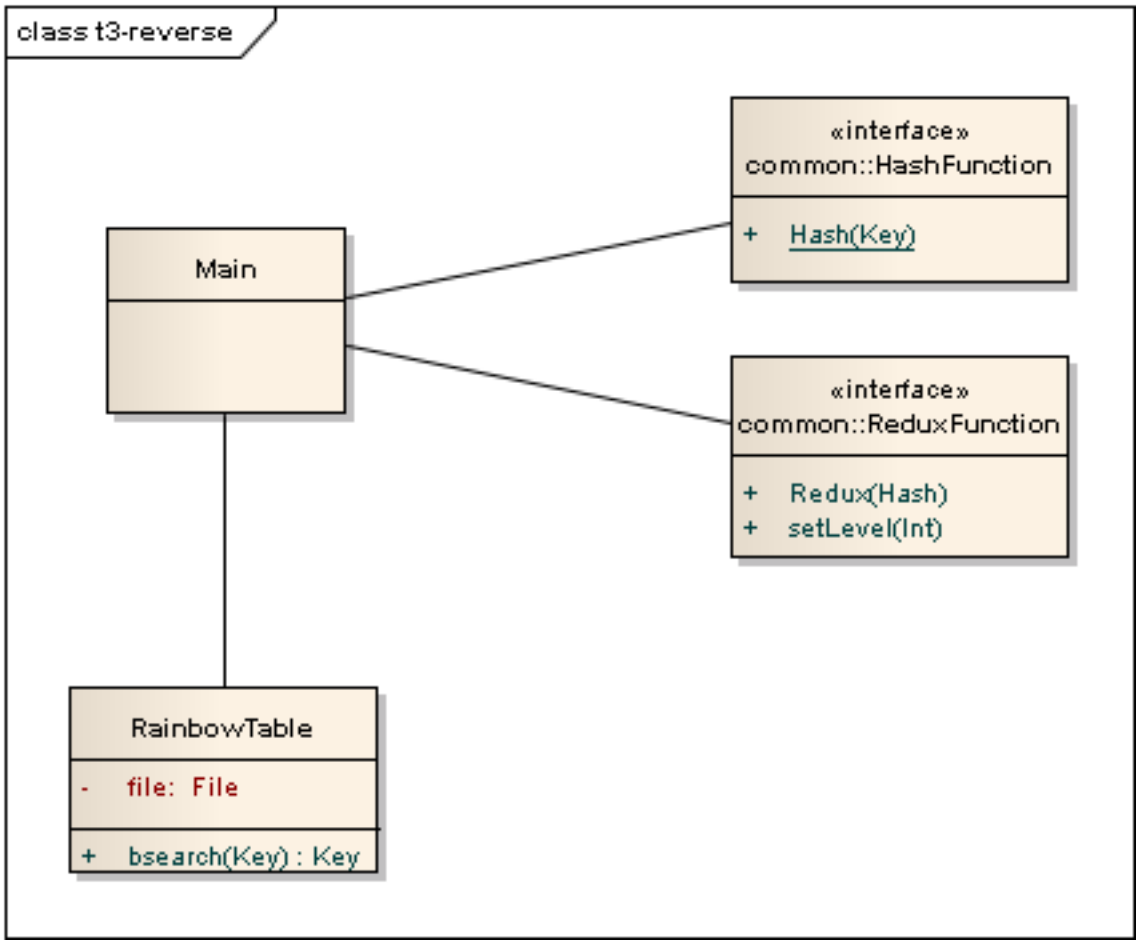
Kiedy w kolejce zostanie tylko jeden plik, zarządca doda do pliku niezbędne nagłówki i program zakończy działanie.

3.1.3 Wyszukiwanie

Program wyszukujący nie będzie podlegał zrównolegleniu, ponieważ jego szybkość będzie w znacznej mierze ograniczona przez prędkość dowolnego dostępu do dysku.

Na początku program wczyta nagłówek tablicy i ustawi odpowiednie wartości długości kluczy, do których były przeprowadzane redukcje, długości łańcuchów i alfabetu. Program będzie obliczał coraz dłuższe łańcuchy na podstawie klucza(zaczynając od łańcucha jednoelementowego) i próbował znaleźć łańcuch o takim samym zakończeniu co obliczone w tablicach.

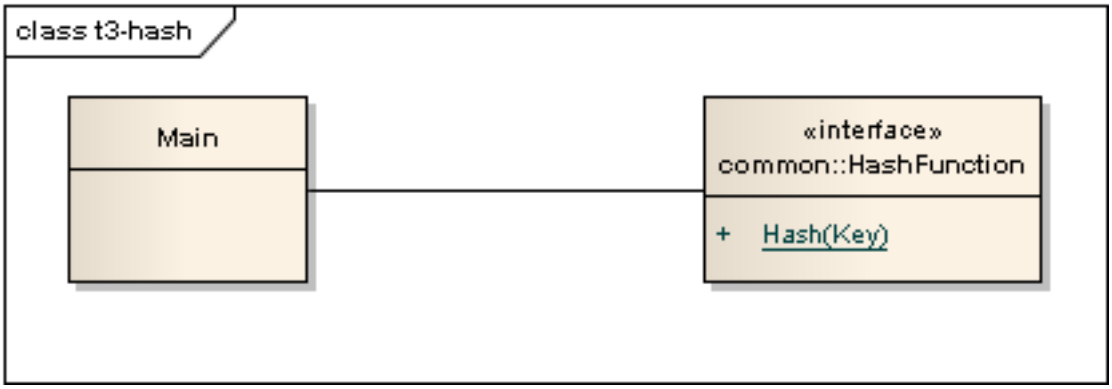
Jeżeli wyszukiwanie się nie powiedzie, program wygeneruje dłuższy łańcuch (chyba,



Rysunek 2: diagram klas programu t3-reverse

że przekroczona zostanie długość łańcucha, dla którego generowana była tablica tęczowa - jeśli tak, program zakończy się niepowodzeniem).

Jeżeli łańcuch znajdujący się w tablicy zostanie dopasowany do obliczonego łańcucha długości n, program wczyta pierwszy element łańcucha i wygeneruje na jego podstawie łańcuch o długości $n - m$ (gdzie m to długość łańcuchów, dla których wygenerowana była tablica). Ostatni element tego łańcucha, to szukany klucz.



Rysunek 3: diagram klas programu t3-hash