Rozproszone łamanie haseł – DES

[w. 1.1]

1 Wprowadzenie

Przedmiotem ćwiczenia jest łamanie prostych haseł zaszyfrowanych jednokierunkową funkcja skrótu (ang. *hash*) realizującą algorytm DES[1].

Ogólna procedura szyfrowania jest następująca:

a) Wybiera się losowy 2-znakowy łańcuch nazywany dalej solą (ang. *salt*).
b) Obliczenie funkcji skrótu wykonywane jest za pomocą funkcji bibliotecznej języka C o nagłówku `char * crypt(const char *patkey, const char *salt).`
c) Zmienna `patkey` zawiera wzorcowe hasło w postaci niezaszyfrowanej, zmienna `salt` zawiera sól.
d) Funkcja `crypt()` zwraca wskaźnik do zaszyfrowanego hasła, które ma długość 13 znaków.
e) Zaszyfrowane wzorcowe hasło przechowuje się do dalszego wykorzystania (np. w celu weryfikacji tożsamości użytkownika).

UWAGA: w haśle wykorzystywane jest tylko 8 pierwszych znaków.

W celu zbadania, czy wprowadzone (np. w trakcie logowania się) hasło jest zgodne z hasłem wzorcowym zapamiętanym w systemie wykonuje się następujące czynności:

a) Badane hasło szyfruje się w sposób opisany wyżej, lecz jako `salt` wprowadza się zaszyfrowane hasło wzorcowe. Jest ono nośnikiem soli użytej wcześniej do zaszyfrowania hasła wzorcowego.
b) Zaszyfrowane hasło badane porównuje się z zaszyfrowanym hasłem wzorcowym.
c) Jeśli wynik porównania jest pozytywny, przyjmuje się, że hasło badane jest identyczne z hasłem wzorcowym (i udziela zezwolenia na dostęp do zasobu lub usługi). W przeciwnym razie, badane hasło traktuje się jako niezgodne z wzorcem (i odmawia takiego zezwolenia).

2 Wymagania dotyczące programu

- Program ma łamać hasła składające się z małych i wielkich liter alfabetu, cyfr i znaków specjalnych (kody ASCII z zakresu 33–126 w reprezentacji dziesiętnej).
- Daną dla programu jest zaszyfrowane hasło wzorcowe wprowadzane jako pierwszy parametr wywołania (`char *zaszyfrowane`).
- Hasło ma być łamane metodą tzw. *brutalnej siły* (ang. *brute force*), lub inaczej – przez systematyczne przeszukiwanie zbioru wszystkich możliwych haseł zbudowanych nad określonym wyżej alfabetem, o maksymalnej długości określonej przez drugi parametr wywołania programu (`int max_dlugosc`).
- Wynikiem łamania hasła jest zbiór wszystkich ciągów znaków, niedłuższych niż wartość zmiennej `max_dlugosc`.
- W metodzie tej generuje się kolejne niezaszyfrowane hasło, oblicza wartość funkcji skrótu i porównuje ją z odrębnie zaszyfrowanym hasłem wzorcowym.
- Zgodność obliczonej wartości z zaszyfrowanym hasłem wzorcowym oznacza, że pasujące hasło zostało znalezione. Następnie cykl jest powtarzany – generowanie jest następne niezaszyfrowane hasło, wykonywane jego szyfrowanie i porównanie z zaszyfrowanym wzorcem. I tak dalej, aż do wyczerpania zbioru możliwych haseł.
- Program winien raportować wszystkie pasujące hasła (w postaci niezaszyfrowanej) oraz łączny czas ich określania.
- W razie nieznalezienia żadnego pasującego hasła, program ma wyprowadzić stosowny napis.
- Łamanie hasła ma odbywać się w sposób zrównoleglony, z wykorzystaniem Open MPI.

[1] Algorytm DES (ang. *Data Encryption Standard*), jako mało bezpieczny, nie jest już używany do szyfrowania haseł

3 Zadanie

a) Opracuj program realizujący określoną wyżej specyfikację. Jego działanie ma być zrównoleglone przy wykorzystaniu API Open MPI.

b) Zbadaj poprawność działania na trzech zaszyfrowanych hasłach ‘a8grQLPu4VyTk’, 'a8nqZPWgNVkw.' oraz ‘a8DeiOhku6L9I’. Obliczenia testowe wykonaj w trybie SMP, korzystając każdorazowo z 1, 2 i 8 procesorów i rejestrując czasy obliczeń (łącznie (3*3=9 uruchomień). Jako maksymalną długość badanych haseł `max_dlugosc` przyjmij 8.

c) Wykonaj łamanie ustalonych przez siebie haseł o długości 1, 2, … i 13 znaków (13 uruchomień). Zarejestruj czasy łamania i oblicz średni czas łamania hasła. Tryb obliczeń MPI (SMP lub rozproszony) wskaże prowadzący zajęcia. Jako maksymalną długość badanych haseł przyjmij 8.

d) Oblicz maksymalną możliwą liczbę cykli (prób) łamania dowolnego hasła nad opisanym wyżej alfabetem, o długości nieznanej algorytmowi.

4 Wskazówki

a) Wskazówki o charakterze ogólnym znajdują się na portalu edukacyjnym w materiałach kursu.

b) Działanie programu A może być naszkicowane w postaci pseudokodu; zmienne o nazwach rozpoczynających się od „`Z_`” dotyczą haseł w postaci zaszyfrowanej.

```
Begin
            Wczytaj(Z_HasloWzorcowe)
            i=0
            Repeat
                  Haslo=Generuj_hasło(i)
                  Z_Haslo=crypt(Haslo, Z_HasloWzorcowe)
                  Zgodne=(Z_Haslo=Z_HasloWzorcowe)
                  i=i+1
            Until (i=liczba_haseł) or Zgodne
            If Zgodne then
                  begin
                        Wyprowadź(Haslo)
                        Wyprowadź(czas_szukania)
                  end
            Else Wyprowadź (komunikat błędu)
End
```

c) Dla ułatwienia implementacji programu przekształć strukturę pseudokodu tak, aby zamiast konstrukcji `repeat-until` wystąpiła pętla `for`.

d) Do skonstruowania programu można wykorzystać program znajdujący się w pliku `login.c`, w katalogu `/mnt/exports/openMPI`. Program ten można również wykorzystać do obliczenia zaszyfrowanego hasła wzorcowego.

e) Kompilacja programu wymaga parametru `-lcrypt`.

5 Opracowanie wyników

Sprawozdanie z realizacji ćwiczenia winno zawierać m.in.:

a) Tekst programu,

b) Czasy łamania różnych haseł oraz średni czas łamania hasła,

c) Obliczenie maksymalnej możliwej liczby cykli łamania hasła.

Do przesłania są 2 pliki:

a) Kompletne sprawozdanie,

b) Kod źródłowy programu do bezpośredniej kompilacji za pomocą gcc/Linux.