Projet GPGPU
Cassage de mots de passe

Contexte

Un administrateur système, voyant que ses utilisateurs n'utilisaient que des mots de passe d'au plus 4 caractères, composés de lettres minuscules et de chiffres uniquement (allez savoir pourquoi...), a décidé de prendre les choses en main et a créé une nouvelle fonction de hachage, sobrement baptisée MD7, pour empêcher les attaquants d'utiliser des outils de cassage automatique, comme hashcat. A vous de lui prouver que ce n'est pas suffisant en codant votre propre outil !

Description de MD7

Message

La fonction de hachage MD7 prend en entrée des messages de longueur arbitraire, auxquels on ajoute des 0 pour obtenir un multiple de 32 bits.

Le message m est ensuite décomposé en n blocs de 32 bits :

m_1	$\cdots$	m_n

Algorithme MD7

L'algorithme MD7 utilise une variable appelée *state* qui représente l'état interne de l'algorithme. Le haché est la valeur de *state* à la fin de l'exécution. L'algorithme est le suivant :

- $state$ = 0xdeadbeef (longueur 32 bits)
- Pour i de 1 à n :
  - $state = state$ XOR m_i
  - $expanded\_state$ = EXPAND($state$)
  - $state$ = COMPRESS($expanded\_state$)

Description de EXPAND

EXPAND est une routine qui étend l'état interne *state* en un tableau de 2^{20} entiers de 32 bits : *expanded_state*. Elle utilise un générateur aléatoire (PRNG) à base de congruences linéaires qui accepte une graine de 64 bits, et donne des valeurs pseudo-aléatoires de 64 bits.

- Pour i de 0 à 1023 :
  - Init_PRNG($i||state$)
  - Pour j de 0 à 1023 :
    - $index = i \times 1024 + j$
    - $expanded\_state[i] = alea()$ XOR $(alea() >> 16)$ XOR $(alea() >> 32)$ XOR $(alea() >> 48)$

Description de COMPRESS

COMPRESS est une routine qui modifie l'état interne *state* à partir du tableau de 2^{20} entiers de 32 bits *expanded_state*. Elle réalise deux étapes de réduction successives.

La première consiste à découper *expanded_state* en blocs de 1024 entiers de 32 bits : ces 1024 entiers sont xorés entre eux pour n'en obtenir qu'un.

Après cette étape il nous reste 1024 entiers que l'on va ajouter entre eux modulo 2^{32} (en utilisant simplement l'opérateur + en C). Cette opération donne un entier qui est le nouvel état interne de l'algorithme.

Votre mission

- Partie séquentielle :

1. Réaliser une implémentation séquentielle de MD7 - des fichiers de référence vous seront fournis pour tester la validité de votre implémentation.
2. Utiliser cette implémentation pour casser un des fichiers de mots de passe fournis. Mesurez le temps nécessaire.

- Partie parallèle :

1. Réaliser une implémentation parallèle de MD7, dans un premier temps en réalisant un haché par bloc.
2. Utiliser une réduction parallèle pour accélérer la routine COMPRESS. Mesurez l'amélioration obtenue pour casser les fichiers de mots de passe.

- Bonus : proposez d'autres optimisations, comme l'utilisation de mémoire partagée, ou la parallélisation de EXPAND.

Une démonstration de votre programme aura lieu en salle de TP. Vous devrez par ailleurs rendre votre code (Makefile apprécié !) ainsi qu'un rapport expliquant brièvement vos choix et vos mesures de performance. Des fichiers de mots de passe différents seront fournis à chaque groupe.