Password Cracking (using Rainbow Tables)
Summary

- Cannot store password in plain text.
- Cryptographic Hash are stored.
- Program to crack hashes using rainbow tables.
- Rainbow Table is a precomputed table of cryptographic hashes.

Why?

- Practical example of space - time tradeoff. (less computer processing time and more storage)
Password Storage

The diagram illustrates the process of password storage:

1. A password (in plaintext) is entered, such as "hello".
2. The password is hashed using a hash function.
3. The hashed password is then stored in a password store.

The hashed password looks like this: $1$r6T8SUB9$Qxe41FyF/3gkPluvKOQ90.
Simplified rainbow table with 3 reduction functions
Design of Sequential Implementation

1. Read No of Hashes to Crack
2. Generate Rainbow Table
3. Password Crack
Sequential Algorithm

Get N hashes to crack from file
For (int i = 0; i < height of Rainbow Table ; i++)
    Generate Chains
        For each i:
            For (int j = 1; j < 10; j++)
                Calculate rainbowTable[i][j]th entry
For(int i = 0; i < N; i++)
    Pass ith hash to crack to RainbowHashCracker
        Search each entry in rainbowTable[height of Rainbow Table][N-1]
        If entry matches with kth element:
            Regen chain at rainbowTable[k][0]
        Return plaintext for which p -> (hash) == ith hash
Design of Parallel Implementation - Password Cracking

ParallelFor()
Hash Division

Core 0
Cracking for Hash_Chunk_0

Core 1
Cracking for Hash_Chunk_1

Core 2
Cracking for Hash_Chunk_2

Core 3
Cracking for Hash_Chunk_3
Parallel Algorithm

Get N hashes to crack from file

ParallelDo

Section 1
ParallelFor (i = 0 to height of Rainbow Table)
  Generate Chains
    For each thread executing some chunk of i to rainbowTable[height]
      For (j = 1 to 10)
        Calculate rainbowTable[i][j]th entry

Section 2
Read Hashes File
ParallelFor (i = 0 to N)
  For each thread executing some chunk of passwords, pass ith hash to crack to RainbowHashCracker
    Search each entry in rainbowTable[height of Rainbow Table][N-1]
    If entry matches with kth element:
      Regen chain at rainbowTable[k][0]
    Return plaintext for which p -> (hash) == ith hash
Demo