**Project report**

In this project, we used the MD5 (Message-Digest Algorithm 5) as our hashing algorithm. MD5 is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. It was designed by Ronald Rivest in 1991 as an improvement of its predecessor, MD4.

The data structure used to create a family tree in this project is a directed graph

**How it works:**

1. **Append Padding Bits:** Padding bits are added to the original message so that the total length of the message is 64 bits less than an exact multiple of 512.
2. **Append Length Bits:** The length bit is added to the output of the first step so that the total number of bits is a perfect multiple of 512.
3. **Initialize MD Buffer:** Four buffers (F,G,H,I) are used, each of size 32 bits.
4. **Process Each 512-bit Block:** A total of 64 operations are performed in 4 rounds. Different functions are applied in each round.

**Pros:**

- Easy to compare: A 32-digit digest is relatively easier to compare when verifying the digests.
- Storing Passwords: Passwords need not be stored in plaintext format, making them inaccessible for hackers and malicious actors.

**Cons:**

- MD5 is prone to length extension attacks.
- It’s an older and insecure algorithm that turns data of random lengths into fixed 128-bit hashes.
- A 2013 attack broke MD5 collision resistance in 2^18 time. This attack runs in less than a second on a regular computer.

- The time and space complexities for the functions used in the MD5 algorithm are …

| Function name | Time complexity | Space complexity |
|---|---|---|
| F,G,H,I | O (1) | O (1) |
| mod_add | O (1) | O (1) |
| left_rotate | O (1) | O (1) |
| round | O (1) | O (1) |
| MD5 | O(n) | O (1) |

- The time and space complexities for functions used in Family tree

| Function name | Time complexity | Space complexity |
|---|---|---|
| add_node | O (1) | O (1) |
| remove_node | O (n) | O (1) |
| find | O (1) | O (1) |
| sibling | O (1) | O (1) |
| is_ancestor | O (logn) | O (1) |
| common_parent | O (n) | O (n) |
| has_relation | O (n) | O (n) |
| furthest_child | O (n) | O (d) |
| __find_root | O (n) | O (1) |
| __bfs | O (n+m) | O(n) |
| find_diameter | O (n+m) | O(n) |

- m= number of edges
- n=number of nodes
- d=depth of tree

The Git repository for this project can be found at:

https://github.com/Tina-Talebi/MysteriousFamily

Authored by: Tina Talebi , Atena Dolati