41900 – Fundamentals of Security
Project Part – 2 (Week 8 – Week 12)

One-Way Property versus Collision-Free Property

In this project you will investigate the difference between hash function's two distinctive properties: one-way property and collision-free property. You are required to use the brute-force method to see how long it takes to break the hash generated using each of these properties. Instead of using `openssl`'s command-line tools, you are required to update the C programs provided to invoke the message digest functions in `openssl`'s crypto library. A sample code can be found at the following link, it would be helpful to familiarize with the sample code.

`http://www.openssl.org/docs/crypto/EVP_DigestInit.html`

Since most of the general hash functions are quite strong against the brute-force attack conducted on those two properties, it can take years to break them using brute-force. Hence, to make the task feasible we reduce the length of the hash value to 24 bits. You can use any one-way hash function, but only the first 24 bits of the hash value generated will be used in this task. As explained, we are using a modified one-way hash function.

"One-way" means that given a function output, you cannot find a matching input, except by trying many potential inputs and getting lucky. A collision is about finding two distinct inputs which yield the same output, without any predefined constraint on said output.

Please design an experiment to find out the following:

1. How many trials it will take you to break the one-way property using the brute-force method? You should repeat your experiment multiple times, and report your average number of trials.
2. How many trials it will take you to break the collision-free property using the brute-force method? Similar to the previous task, report the average of multiple attempts.
3. Based on your observation, describe which property is easier to break using the brute-force method and why?

There are two skeleton programs i.e. `oneway.c` and `collision.c` provided foe this assignment. You aim is to complete the code, run it successfully and observe which property is easier to break using the brute-force method.

The character set to be used is as follows:

0123456789!\"#$%&'()*+-/. ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

41900 – Fundamentals of Security
Project Part – 2 (Week 8 – Week 12)

1. Brute-Force One-Way property

The program's objective is to hit a "target" hash value by generating the hash for different strings and matching it with the target hash value.

The approach to generate strings and match them against a target is implemented as follows:

1. Start with string length 1
2. Generate all possible combinations of strings for current length using the character set, generate its hash and compare it with the target hash.
3. If a match is found, the execution stops and reports the number of iterations used.
4. If no match is found in all possible strings of current length, increment length by '1' and go to step (2)

The brute force function is used in the one way property in `oneway.c` to generate strings. The program follows a systematic manner by exhausting all possibilities for a given length (starting with 1) and incrementing the length if no match was found. You can refer to `oneway.c` program for 77 characters that can be used for to brute-force one way property.

The `oneway.c` should be executed a minimum of five times to find out the average number of tries required to match the target hash.

2. Brute-Force Collision-Free property

In the case of collision free property, there is no restriction on target hash. The approach followed in the `collision.c` program to break this property is as follows:

1. Allocated an array for saving hash values
2. Generate initial message (call it M)
3. Find M's hash
4. Generate random string and find its hash
5. Compare hash from (4) to M's hash, if a match is found then exit and report the number of iterations used.
6. Compare hash from (4) to each hash value in array, if any match is found then exit and report the number of iterations used.
7. If no match is found, save the hash from (4) into the array and go to step (4)

To break the collision free property, you just need to find two messages with matching hash values. Matching the random string's hash value to an initial hash as well as saved hash value (in the array) increases the probability of finding hash value with less number of iterations.

Execute the `collision.c` program five times to find the average number of iteration required.

41900 – Fundamentals of Security
Project Part – 2 (Week 8 – Week 12)

INSTRUCTIONS FOR COMPILING THE CODE

In order to compile and run the code, please use the command as shown below:

To compile:

```
$ gcc –o oneway oneway.c –lcrypto –ldl –std=c99
$ gcc –o collision collision.c –lcrypto –ldl –std=c99
```

To run the code:

```
$./oneway md5
$./collision md5
```