Juan Pinzon
SID: 23632316
Section: 104. TA: Arjun
CS161 – Computer Security - HW4

HW4 Write up

- **gentable.c**

gentable.c is in charge of generate the rainbow table using s, n as parameters. Include the following functions: **maxNumLines** which return the maximum number of entries on rainbow file, so it satisfies the size restriction 2^s; **getNextAvailableKey** which return the lowest-value key that haven't seen before; **getKey_128** which receives a 32-bit key and return a 128-bit key and **reduceKey** which is the Reduce function based on the formula R(h,i) = (h + (i % k)) % 2^n, I choose k=3 cause gave me better results. I use a two bitmap variables to keep track of the keys: **key_bitmap** which keeps track of all generated keys to avoid collisions on a chain Key→Hash→Reduce→Hash…. and **aes_key_bitmap** which keeps track only of the first keys on a chain. The rainbow file entries consist of a tuple [key, $AES_{key}(0)$], that use the struct **rainbow_data** to write on the binary file rainbow.

The engine of my code consist on a **main loop** that **ends** when the file size restrictions 3*16*2^s bytes is reached **OR** when the maximum number of passwords 2^n-1 in **aes_key_bitmap** is reached. Each iteration of the loop the program execute a chain Key→Hash→Reduce→Hash keeping track of all generated keys using **key_bitmap**. The chain is break whenever is a collision with a previous key. To avoid stopping all my chains because they have collided and there is no more keys available, I use a Reduce function that varies depending on where in a chain I am. Thus, using this equation R(h,i) = (h + (i % k)) % 2^n I get more variety of keys and less collisions than with a naïve reduce function. After the chain finishes, the first key on the chain is stored on **aes_keys_bitmap**, and a new entry [key, $AES_{key}(0)$] is added to the rainbow table. Even though I use a more sophisticated Reduce function there were still collision, so I did not reach the maximum number of passwords 2^n -1, even when n=s, that means I have plenty of space and could pre-compute all possible hashes, so in crack.c I could find the key in nearly constant time. So I implemented a mechanism where inside the main loop I check if the **key_bitmap** is full, if it is full then copy **aes_key_bitmap** (keep track only of the first keys on the chain) on the **key_bitmap.** This guarantee I use all the space available while I get new keys.

- **crack.c**

crack.c is in charge of read the rainbow file generated on gentable.c and find a match with the ciphertext entered as an argument. . Include the following main functions: **check_rainbow** which traverse the rainbow file looking for a match with the ciphertext. Return true if there is a match, false otherwise; **getCiphertext** which receives a char array (string enter in arguments 0x...) and return an unsigend char array with the

ciphertext value; and **compareCiphertext** which compare two Ciphertext.. If both are equals return true, otherwise false. The core of **crack.c** is simple: I read the file and check if there is a match between each entry (only the hash part) on the file and the ciphertext entered as argument. At the same time I store the keys in a bitmap **key_bitmap** to keep track of the current keys known. If I **found a match** on the **rainbow** file then I **reproduce the chain** from the first key, until I find the ciphertext. If the ciphertext is not in the rainbow file then I started computing AES, with the difference I already know the keys on the rainbow, so I look for the next available key on **key_bitmap**. This strategy allowed me to inverse all the password challenges, however the number of AES evaluations increases drastically as s reduces, until I start running into birthday paradox collision problems(n>2*s). Likewise I noticed that the number of expected AES evaluation by crack increases drastically when n – s >=4.

- 20-bit password with hash: 0xae60abdcb19d5f962a891044129d56d4

Password is 0x31415. AES was evaluated 1 times.

- 24-bit password with hash: 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 1 times.

- 28-bit password with hash: 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 1 times.

Juan Pinzon
SID: 23632316
Section: 104. TA: Arjun
CS161 – Computer Security - HW4

TEST RESULTS

./gentable 20 20

./crack 20 20 0xae60abdcb19d5f962a891044129d56d4

Password is 0x31415. AES was evaluated 1 times.

./gentable 20 19

./crack 20 19 0xae60abdcb19d5f962a891044129d56d4

Password is 0x31415. AES was evaluated 1 times.

./gentable 20 18

./crack 20 18 0xae60abdcb19d5f962a891044129d56d4

Password is 0x31415. AES was evaluated 1 times.

./gentable 20 17

./crack 20 17 0xae60abdcb19d5f962a891044129d56d4

Password is 0x31415. AES was evaluated 56497 times.

./gentable 20 16

./crack 20 16 0xae60abdcb19d5f962a891044129d56d4

Password is 0x31415. AES was evaluated 56497 times.

./gentable 24 24

./crack 24 24 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 1 times.

./gentable 24 23

./crack 24 23 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 1 times.

./gentable 24 22

./crack 24 22 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 1 times.

./gentable 24 21

./crack 24 21 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 11672124 times.

./gentable 24 20

./crack 24 20 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 14188706 times.

./gentable 24 19

./crack 24 19 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 15446997 times.

./gentable 24 18

./crack 24 18 0xeb94f00c506705017ce61273667a0952

Password is 0xFEE707. AES was evaluated 16076143 times.

./gentable 28 28

./crack 28 28 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 1 times.

./gentable 28 27

./crack 28 27 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 1 times.

./gentable 28 26

./crack 28 26 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 1 times.

./gentable 28 25

./crack 28 25 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 1 times.

./gentable 28 24

./crack 28 24 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 1 times.

./gentable 28 22

./crack 28 22 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 23572772 times.

./gentable 28 21

./crack 28 21 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 28605937 times.

./gentable 28 20

./crack 28 20 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 31122519 times.

./gentable 28 19

./crack 28 19 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 32380810 times.

./gentable 28 18

./crack 28 18 0xa2cf3f9d2e3000c5addea2d613acfda8

Password is 0x2014ABC. AES was evaluated 33009956 times.