IIS P1a – Cryptochallenge
–

Group 29
Thomas Hödl, Mario Theuermann, Stephan Valentan, Camilla Reis

Specifications:
Just start programs „factoringOfN.py“ (Task 1) and „collision_search.py“ (Task 2) with Python 3.5; no parameters needed.

Task1:

Summary of code solution:
The main function of the program is the function decrypt. This function evokes the function factorising at the beginning.
The function factorising is responsible for determining one prime number of the given n. This function uses the Pollard's Rho Algorithm for calculating one prime number of n.

After the function factorising finished and we got one prime number, the function decrypt calls the function generatePrivateKey to generate the appropriate private key to the given public key e. The function generatePrivateKey uses the RSA algorithm to create the private key. At the end of the function generatePrivateKey there is used the extended euclid algorithm to generate the private key d with the public key e and phi of n.

After we got the private key from the function generatePrivateKey, the main function decrypt encodes the given encrypted message. The result of this encoding is a number. This number is decrypted with the RSA algorithm and the previous generated private key. The result of the decryption is another number. This number is decoded to a text which is the decrypted message.

Algorithms:

Pollard‘s Rho algorithm
https://www.cs.colorado.edu/~srirams/courses/csci2824-spr14/pollardsRho.html

Ext. Euclid algorithm:
https://en.wikibooks.org/wiki/Algorithm_Implementation/Mathematics/Extended_Euclidean_algorithm

RSA:
http://doctrina.org/How-RSA-Works-With-Examples.html

Given:

```
e:          867512337310254731119
n:          1180592782282757817137
c:          "MMCTNLAZNXEFUEN"
```

Solutions:

```
q:          34359771223
p:          34359739319
phi:        1180592782214038306596
d:          42006330222808060219
Message:    KAPFENBERG


Runtime in seconds:     ~ 0.853201150894165
Memory requirements:    ~ 3,2 MB
```

Task2:

Summary of code solution:

The program has two functions. The first one is the function SHA2mod. This function is the hash function. It meets the requirements of the task description.

The second function is the function searchCollision. This function detects the collision. It uses Brent's algorithm for it.
So, the function consists of 3 parts. The first part contains a while loop that determines that there exists a circle. This while loop also calculates the circle length.
The second part consists of a for loop. This for loop lets the hare run until it reaches the circle length. Before the for loop the hare and the tortoise are set to the start.
The last part is a while loop. The hare and the tortoise “run” with the same speed until they collide within the while loop. They will collide because they are exactly the circle length apart from each other.

Algorithms:

Brents algorithm for circle detection:
https://en.wikipedia.org/wiki/Cycle_detection

Detecting a Loop:
http://codingfreak.blogspot.com/2012/09/detecting-loop-in-singly-linked-list_22.html
https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=55825

Given:

```
Prefix:      1430320143050014305291430751

Started with number:    2ac5 (randomly chosen)
```

Solutions:

```
Collision with hash value df62fa19cc3ac958

Colliding values (prefix + hash value):
    14303201430500143052914307512f1a142ddfa25079
    1430320143050014305291430751d71c3c584b5bf4d3



Runtime in hours:          ~ 24h
Memory requirements:       ~ 3,3 MB
```