RSA Quest

July 19, 2020

This notebook will walk you through the cryptographic quest we've created for you!

You're going to solve a series of puzzles. First few of them will lead you to a secret link containing more puzzles and instructions, then you'll get another link, and so on. In the end, you will get the final answer. You'll be able to get points for the course for solving some parts of this quest. To do so, you will need to submit the information you get in this notebook to the next quiz called "RSA Quest Quiz".

You won't need to do any programming or guesswork in this notebook. If you've solved some of the questions in the previous quiz, "RSA Quiz", you'll need to copy the code of your solutions to the corresponding functions in this notebook, launch the corresponding parts of the notebook and get the keys. You will be able to get some points even if you didn't solve all the questions of the "RSA Quiz"!And the first puzzle is based on the question 2 of the "RSA Quiz". In the next part of the notebook, first go some utility functions, then goes the main function "Decrypt". Please copy the code of your correct implementation of the function Decrypt in that question so as to substitute our current implementation, then launch the corresponding parts of the notebook to get the first part of secret link.

```
In [ ]: import sys, threading


        sys.setrecursionlimit(10**7)
        threading.stack_size(2**27)


        def ConvertToInt(message_str):
          res = 0
          for i in range(len(message_str)):
            res = res * 256 + ord(message_str[i])
          return res

        def ConvertToStr(n):
            res = ""
            while n > 0:
                res += chr(n % 256)
                n //= 256
            return res[::-1]

        def PowMod(a, n, mod):
            if n == 0:
```

```
            return 1 % mod
        elif n == 1:
            return a % mod
        else:
            b = PowMod(a, n // 2, mod)
            b = b * b % mod
            if n % 2 == 0:
              return b
            else:
              return b * a % mod

    def ExtendedEuclid(a, b):
        if b == 0:
            return (1, 0)
        (x, y) = ExtendedEuclid(b, a % b)
        k = a // b
        return (y, x - k * y)

    def InvertModulo(a, n):
        (b, x) = ExtendedEuclid(a, n)
        if b < 0:
            b = (b % n + n) % n
        return b

    def Decrypt(ciphertext, p, q, exponent):
      # Substitute this implementation with your code from question 2 of the "RSA Quiz".
      return ConvertToStr(PowMod(ciphertext, exponent, p * q))

    p = 779849711281
    q = 748173698927
    e = 1018651
    ciphertext = 148784435264686331994392
    decrypt_first_puzzle = Decrypt(ciphertext, p, q, e)
    print(decrypt_first_puzzle)
```

The second puzzle is based on the questions 1 and 3 of the "RSA Quiz". Here, you need to substitute our implementation of the function "Encrypt" with your implementation from question 1 and substitute our implementation of the function "DecipherSimple" with your implementation from the question 3 of the "RSA Quiz" to get the second part of the secret link.

```
In [ ]: def Encrypt(message, modulo, exponent):
          # Substitute this implementation with your code from question 1 of the "RSA Quiz".
          return PowMod(ConvertToInt(message), 2, 4)

        def DecipherSimple(ciphertext, modulo, exponent, potential_messages):
          # Substitute this implementation with your code from question 3 of the "RSA Quiz".
          if ciphertext == Encrypt(potential_messages[0], modulo, exponent):
            return potential_messages[0]
          return "don't know"
```

```
         ciphertext = 3361840230471186770867 39
         modulo = 1110014195838866450995043
         exponent = 767549
         potential_messages = ["http://goo.gl/", "http://tinyurl.com/", "http://bit.ly/", "http
         decrypt_second_puzzle = DecipherSimple(ciphertext, modulo, exponent, potential_message
         print decrypt_second_puzzle
```

Now you're ready to construct the secret link. Just run the next part of the notebook to get it. Then follow the link to get more instructions. Also, if you follow the correct link, you will already get the answer to the first question of the following "RSA Quest Quiz"!

```
In [ ]: secret_link = decrypt_second_puzzle + decrypt_first_puzzle
        print(secret_link)
```

Substitute the implementation of the function "DecipherSmallPrime" below with your implementation from "RSA Quiz", question 4. Also, insert the values for the ciphertext, modulo and exponent from the secret document. Then launch to get the first part of the next secret link.

```
In [ ]: def DecipherSmallPrime(ciphertext, modulo, exponent):
          # Substitute this implementation with your code from question 4 of the "RSA Quiz".
          return ConvertToStr(PowMod(ciphertext, exponent, p * q))


        ciphertext = 1
        modulo = 100000000000000000
        exponent = 1
        decrypt_third_puzzle = DecipherSmallPrime(ciphertext, modulo, exponent)
        print(decrypt_third_puzzle)
```

Substitute the implementation of the function "DecipherSmallDiff" below with your implementation from "RSA Quiz", question 5. Also, insert the values for the ciphertext, modulo and exponent from the secret document. Then launch to get the second part of the next secret link.

```
In [ ]: def IntSqrt(n):
          low = 1
          high = n
          iterations = 0
          while low < high and iterations < 5000:
            iterations += 1
            mid = (low + high + 1) // 2
            if mid * mid <= n:
              low = mid
            else:
              high = mid - 1
          return low

        def DecipherSmallDiff(ciphertext, modulo, exponent):
          # Substitute this implementation with your code from question 5 of the "RSA Quiz".
          small_prime = IntSqrt(modulo)
```

```
  big_prime = modulo // small_prime
  return Decrypt(ciphertext, small_prime, big_prime, exponent)

ciphertext = 1
modulo = 10000000000000000000
exponent = 1
decrypt_fourth_puzzle = DecipherSmallDiff(ciphertext, modulo, exponent)
print(decrypt_fourth_puzzle)
```

Now you have both parts of the second link. To get the link, just launch the next part of the notebook. Then follow the link to get more instructions. Also, if you follow the correct link, you will get the answer to the second question of the following "RSA Quest Quiz"!

```
In [ ]: second_secret_link = decrypt_third_puzzle + decrypt_fourth_puzzle
        print(second_secret_link)
```

Substitute the implementation of the function "DecipherCommonDivisor" below with your implementation from "RSA Quiz", question 6. Also, insert the corresponding values for the ciphertexts, modulos and exponents from the second secret document. Then launch to get the first and the second parts of the final answer.

```
In [ ]: def GCD(a, b):
          if b == 0:
            return a
          return GCD(b, a % b)

        def DecipherCommonDivisor(first_ciphertext, first_modulo, first_exponent, second_ciphe
          # Substitute this implementation with your code from question 6 of the "RSA Quiz".
          for common_prime in range(2, 1000000):
            if first_modulo % common_prime == 0 and second_modulo % common_prime == 0:
              q1 = first_modulo // common_prime
              q2 = second_modulo // common_prime
              return (Decrypt(first_ciphertext, common_prime, q1, first_exponent), Decrypt(sec
          return ("unknown message 1", "unknown message 2")

        first_ciphertext = 1
        first_modulo = 1000000000000000
        first_exponent = 1
        second_ciphertext = 1
        second_modulo = 9999999999999999
        second_exponent = 1

        decrypt_sixth_puzzle = DecipherCommonDivisor(first_ciphertext, first_modulo, first_expo
        print(decrypt_sixth_puzzle)
```

Substitute the implementation of the function "DecipherHastad" below with your implementation from "RSA Quiz", question 7. Also, insert the values for the ciphertexts and modulos from the second secret document. Then launch to get the third - and the last - part of the final answer.

```
In [ ]: def ChineseRemainderTheorem(n1, r1, n2, r2):
          (x, y) = ExtendedEuclid(n1, n2)
```

```
      return ((r2 * x * n1 + r1 * y * n2) % (n1 * n2) + (n1 * n2)) % (n1 * n2)

    def DecipherHastad(first_ciphertext, first_modulo, second_ciphertext, second_modulo):
      # Substitute this implementation with your code from question 7 of the "RSA Quiz".
      r = ChineseRemainderTheorem(first_modulo, first_ciphertext, second_modulo, second_ci
      return ConvertToStr(IntSqrt(first_ciphertext * second_ciphertext))

    first_ciphertext = 1
    first_modulo = 1000000000000000
    second_ciphertext = 1
    second_modulo = 999999999999999

    decrypt_seventh_puzzle = DecipherHastad(first_ciphertext, first_modulo, second_ciphert
    print(decrypt_seventh_puzzle)
```

Now just launch the next part to get the final answer, then copy and paste to submit it as the answer to the question 3 of the "RSA Quest Quiz"!

```
In [ ]: final_answer = decrypt_sixth_puzzle[0] + decrypt_sixth_puzzle[1] + decrypt_seventh_puz
        print(final_answer)
```