HYBRID ENCRYPTION I

A Project Report

Presented to

Prof. Auston Davis

Department of Computer Science

San José State University

In Partial Fulfilment

Of the Requirements for the Class

CS265

By

Akhil Kumar Gour

Avinash Nanjappan

October 2017

TABLE OF CONTENTS

I. Problem Statement

The purpose of encryption algorithms is to encrypt the plaintext or the message and generate ciphertext which can only be decrypted by the intended recipient. Symmetric and asymmetric are two groups of encryption algorithms. In symmetric encryption schemes, both the sender and the receiver share the same key for encrypting and decrypting the data. This scheme provides advantages in term of performance and speed, the trade-off being security. Its biggest drawback is the problem of key exchange. The sender and receiver must exchange a secret key in a secure way before encrypting and sending the data. Asymmetric encryption algorithms solve this problem by using a pair of public and private key. Its drawback is that the cost of data encryption is very high.

Using hybrid encryption, one takes advantage of both of these algorithms as it incorporates the simplicity of symmetric cipher with the almost unbreakable security features of asymmetric cipher. For encrypting the data, a random key K is generated with a symmetric scheme, Advanced Encryption Standard (AES), using which, plaintext is encrypted to ciphertext and then an asymmetric encryption scheme, Rivest–Shamir–Adleman (RSA), is used to encrypt and transfer this key K which can only be decrypted by the intended recipient. This approach presents a faster and much secure way for encryption and decryption.

II. Challenge

- You eavesdropped on a hybrid encryption communication.
- You know that the asymmetric scheme RSA is used in its plain form, i.e. the session key is encrypted as $c = K^e Mod\ N$.
- The session key itself is used in an 128 bit AES Cipher in ECB mode to encrypt the plaintext message.
- Find the plaintext message.

In the additional file you will find the public RSA parameters (N, e) and the ciphertexts. The encryption of the session key under RSA is given as C_p and the encryption of the plaintext under AES using the session key is given as C_s.

III. Additional Details

C_s = 0xfd0b934c23288975648cd1d03ed3c5e2

C_p =
0xc0eacf32dc0492464d9616fefc3d01f56589a137781bf6cf56784dea1c44ef52d61b1025655f370eb78646716f93e0a5

n =
0x9c5f36caf9adc60b4447897c639f1564ed0709251147276de030db395555c8afed912a198b334bd2301981731282981 26e958e38cac653e061035e300505eed1

e = 0x3

IV. Implementation of the Cryptosystem

- For implementing the hybrid encryption system, we have to implement two algorithms- **AES and RSA**.
- **AES -**
    1. By AES, we encrypt the plaintext message that we want to transfer with the help of a key.
    2. We will denote the plaintext message as **M**, ciphertext as **C_s**, and the key as **K**.
    3. The key is generated using Rijndael's key schedule.
    4. For encrypting message M with key K, there are 4 operations involved:
        a. **Key Expansions**
        b. **Initial Round: Add Round Key**
        c. **Rounds:**
            i. SubBytes
            ii. ShiftRows
            iii. MixColumns
            iv. AddRoundKey

d. Final Round (no MixColumn)

i. SubBytes

ii. ShiftRows

iii. AddRoundKey

5. After these operations, we get **C_s**, which is the ciphertext or the message in the encrypted format.

- **RSA –**

1. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at much higher speed.
   [Source: https://en.wikipedia.org/wiki/RSA_(cryptosystem]
2. Given the values of N and e (which are public), for encrypting the **session key K** or the message M, we use the following formula:

$$c = K^e Mod\ N$$

3. Here,
   a) **K** is the session key, which was used for encrypting data during AES, and which needs to be encrypted by RSA. It is the plaintext message for RSA.
   b) **N** is a relatively large number derived as a product of 2 relatively prime numbers **p** and **q**.
   c) We calculate λ(n), which is (p-1)(q-1).
   d) **e** is an integer such that:
   - $1 < e < \lambda(n)$, and
   - $\gcd(e, \lambda(n)) = 1$;
     i.e., e and $\lambda(n)$ are coprime.

   e) Find **d** such that ed = 1 mod (p-1) (q-1).
   f) Public key is (N,e).
   g) Private key is d.
4. To encrypt **M**, we compute:

$$c = K^e Mod\ N$$

5. To decrypt ciphertext **C**, we compute:

$$K = c^d Mod\ N$$

V. Attack on the System

- In the given Hybrid Cryptosystem, the original message M is encrypted by AES scheme, with the help of a key K. This key K is then encrypted using RSA and sent to the recipient.
- The objective of an attack on this system is to get the decrypted key i.e. K (which was encrypted in RSA) and use this K to decrypt the message M (which is the original message M in AES).
- The attack is carried out in 2 phases:

PHASE I: Obtain the key K from RSA.

Encryption in RSA is done by:

$$C\_p = K^e Mod\ N$$

The values known to us are C_p, e and N.

With the given information, the attacks that are possible on this encryption process are:

- **Chosen Plaintext Attack:** Since RSA has no random component, an attacker can successfully launch a chosen plaintext attack against the cryptosystem, by encrypting likely plaintexts under the public key and test if they are equal to the ciphertext.

- **Chosen Ciphertext Attack:** RSA has the property that the product of two ciphertexts is equal to the encryption of the product of the respective plaintexts. That is $m_1^e m_2^e \equiv (m_1 m_2)^e \pmod{n}$. Because of this multiplicative property a chosen-ciphertext attack is possible.

- **Chinese remainder theorem:** If the same clear text message is sent to e or more recipients in an encrypted way and the receivers share the same exponent e, but different p, q, and therefore n, then it is easy to decrypt the original clear text message via the Chinese remainder theorem.

- When encrypting with low encryption exponents (e.g., e = 3) and small values of the m, (i.e., $m < n^{1/e}$) the result of m^e is strictly less than the modulus n. In this case, ciphertexts can be easily decrypted by taking the *e*th root of the ciphertext over the integers.
  [Source: https://en.wikipedia.org/wiki/RSA_(cryptosystem)]

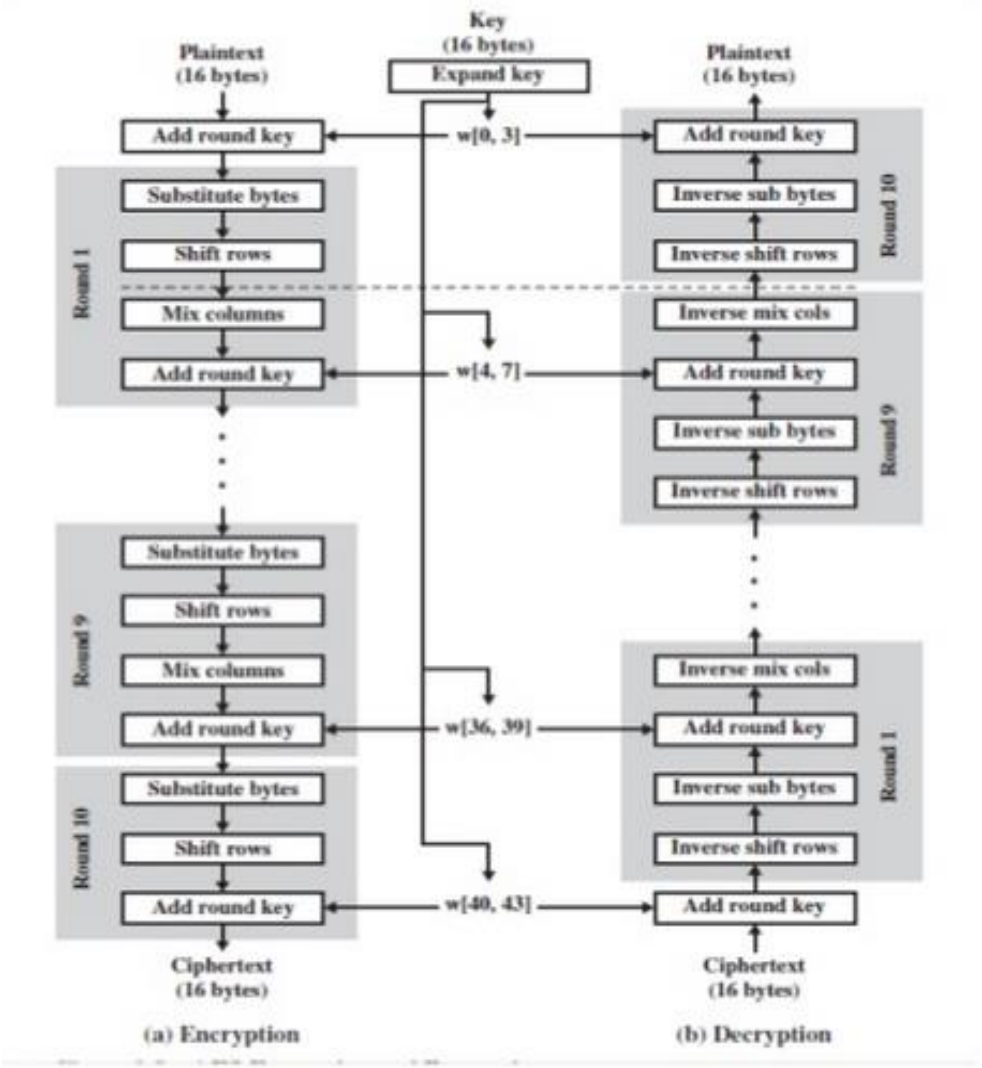
Another way to get **K** is by using the decryption formula:

$$K = c^d Mod\ N$$

- The values known to us are N and c.
- In order to obtain K, we must find the value of d (the private key).
- d is obtained by using the formula: ed = 1 mod (p-1) (q-1).
- Hence, anyone who has d, the private key, can decrypt and get K.

PHASE II: Use this key K to get the plaintext from AES.

- There are four operations in AES Decryption. These are exact reverse of operations of Encryption.
- They are,
  - Inverse SubBytes
  - Inverse ShiftRows
  - Inverse AddRoundKey
  - Inverse MixColumns
- After finishing these steps, we get the plaintext message **M**.
- The structure of AES is given in the figure below:



Structure of AES

VI. Work Factor

The work factor required for this project is given as a progression timeline:

1. **Week 1: 11 September -18 September**
    - Studied the cryptosystem
    - Understood what Hybrid Encryption is and how it works.
    - Understood AES and its different types i.e. ECB, CBC and CTR mode.
    - Formalized and planned a way to decrypt the ciphertext we had C_s, which was encrypted using a key K, in a 128-bit AES cipher in ECB mode.
    - Started implementing software to encrypt a plaintext message using 128-bit AES cipher.
    - RSA was taught the following week

2. Week 2: 18 September - 24 September

- Learnt how RSA works.
- We understood how final cryptosystem would work i.e. we got the idea of the flow of the whole system.
- Which value is associated with which variable and what values will be transformed to and from one algorithm to the other.
- We figured out that the session key was used to encrypt original message M and convert it into ciphertext c-s with AES and then this key is transferred to RSA for encryption which yields up ciphertext c-p.
- Using the formula for decrypting the ciphertext k=c-p Mod N, we thought of finding d to get the plaintext.
- To get d, the private key, it was a big task.
- d is obtained by using this formula

$$ed = 1\ mod\ ((p-1)\ *(q-1))$$

where,

- e is an integer which is relatively prime to (p-1)(q-1).
- p and q are factors of N, which are relatively prime.
- So, we had to factorize N to get d.
- The N given to us was a 512-bit (154 digit) number.
- RSA was designed to be a one-way function i.e. to get N by multiplying p and q was easy but getting p and q from N was not.
- Still, we wrote a java program to get the prime factors of N.
- N was to be entered by the user and was stored in a Big Integer variable.
- It was a simple factorization method, so for small value of N it gives the output instantly but for large N, it was taking too long.
- So, we studied and tried using different factorization methods like quadratic sieve, Pollard's number field sieve, Fermat's factorization method and general number field sieve.

Week 3: 25 September – 1 October

- Running the program for these algorithms on our system didn't give output in reasonable time (we ran these programs for 16 hours straight or even 24 hours sometimes, but in vein).
- So, we decided to switch to high performance open-source mathematics software system SageMath, which included many open-source packages like NumPy, SciPy etc.
- We ran our program on SageMath for more than 20 hours and still we didn't get an output.
- Parallel to this, we started writing the code for RSA for our hybrid cryptosystem program.
- Towards the end of the week, we concluded that if was not possible to easily find p and q(we based our conclusions by reading few blogs and said it might take around a couple of month to crack the RSA by factorizing a 512-bit key.

Week 4: - 1 October – 8 October

- Started looking at other alternatives instead of trying to factorize N.
- Came across a few attacks that are possible on RSA.
- Studied and tried to implement the following attacks: -
    - Chosen plaintext
    - Chosen ciphertext
    - Chinese Remainder theorem
    - Cube root attack
- The efficiency RSA in term of security is almost dependent on the value of e.
- Since e given to us was 3, we decided to go with a version of cube root attack.
- If e = 3,
    - Then $C\_p = k^3$.
    - Which is $C_p^{\left(\frac{1}{3}\right)} = K$.

- i.e. the key used for encrypting the plaintext message M, is the same as the cube root of c_p.
- Therefore,
  - Given N = 0x9c5f36caf9adc60b4447897c639f1564ed0709251147276de030db395555c8afed912a198b334bd2301981731282981 26e958e38cac653e061035e300505eed1
  - Cp = 0xc0eacf32dc0492464d9616fefc3d01f56589a137781bf6cf56784dea1c44ef52d61b1025655f370eb78646716f93e0a5
  - E = 3
  - $K = C_p^{\left(\frac{1}{3}\right)}$, which comes out to be,

    = 309658584779820310739729975902632468029
- Passed this key to our AES program which carried out the necessary functions and gave us the plaintext as Diffie Rocks !!!
- Entered this on the challenge and entered the Hall of Fame.

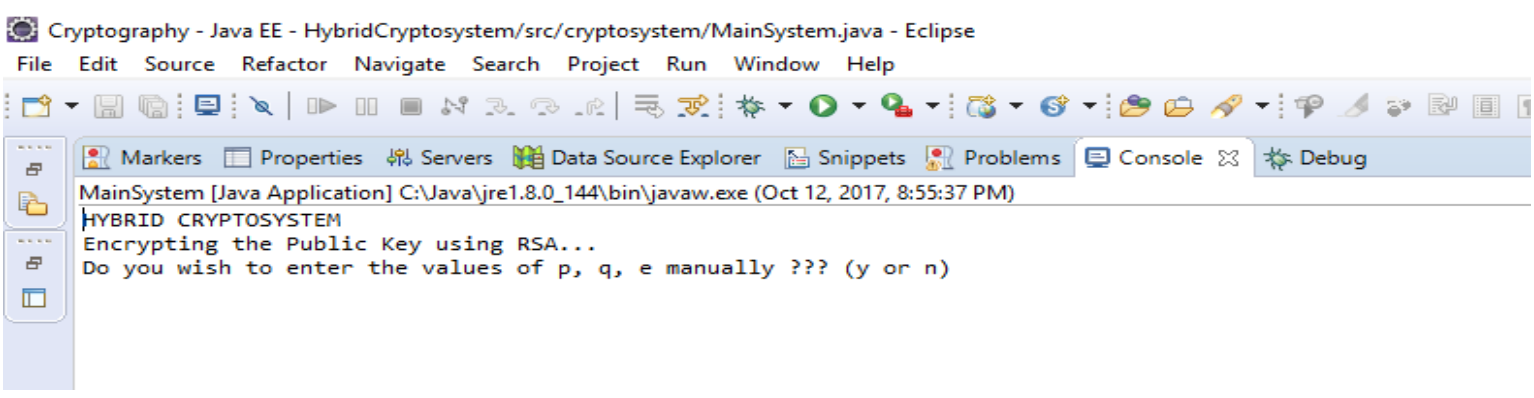
Week 5: 8 October – 12 October

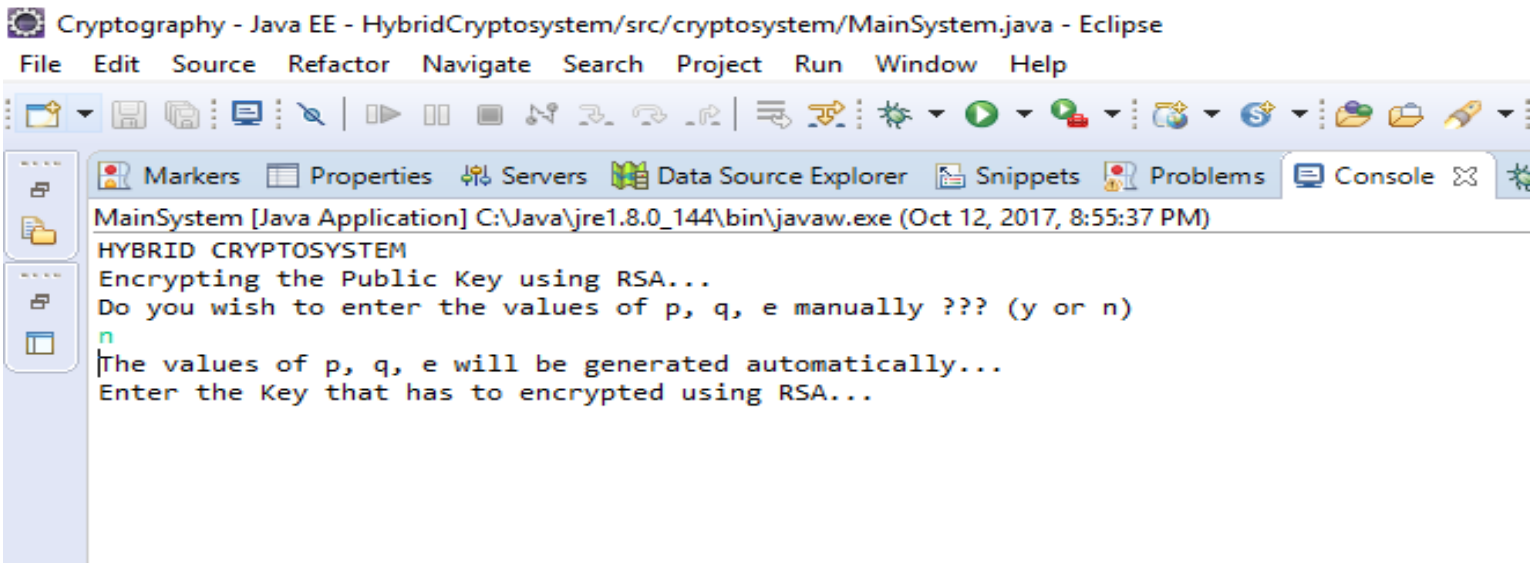
- Preparing the report.
- Organizing the structure of the code and performing the optimizations.
- Provided comments in the code at necessary places.

VII. Computational Experiments

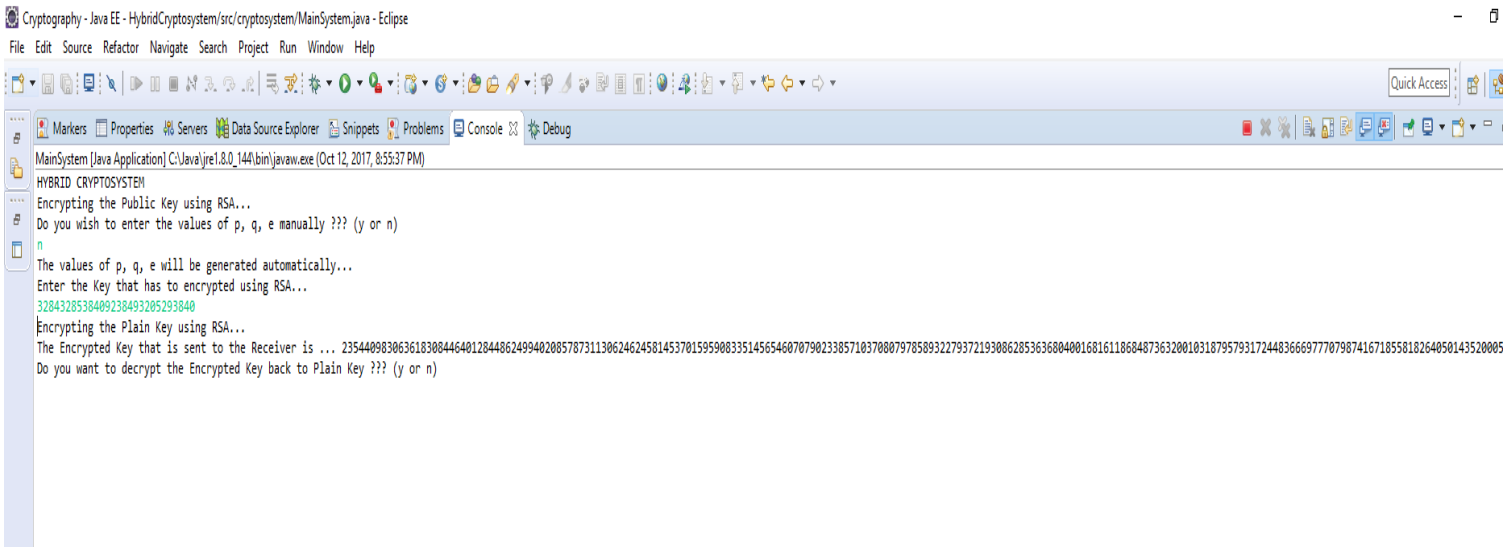
Part I: Output of Implementation of the system:

The program first asks the user whether he wishes to enter the values of p, q and e manually or if it has to be generated automatically.

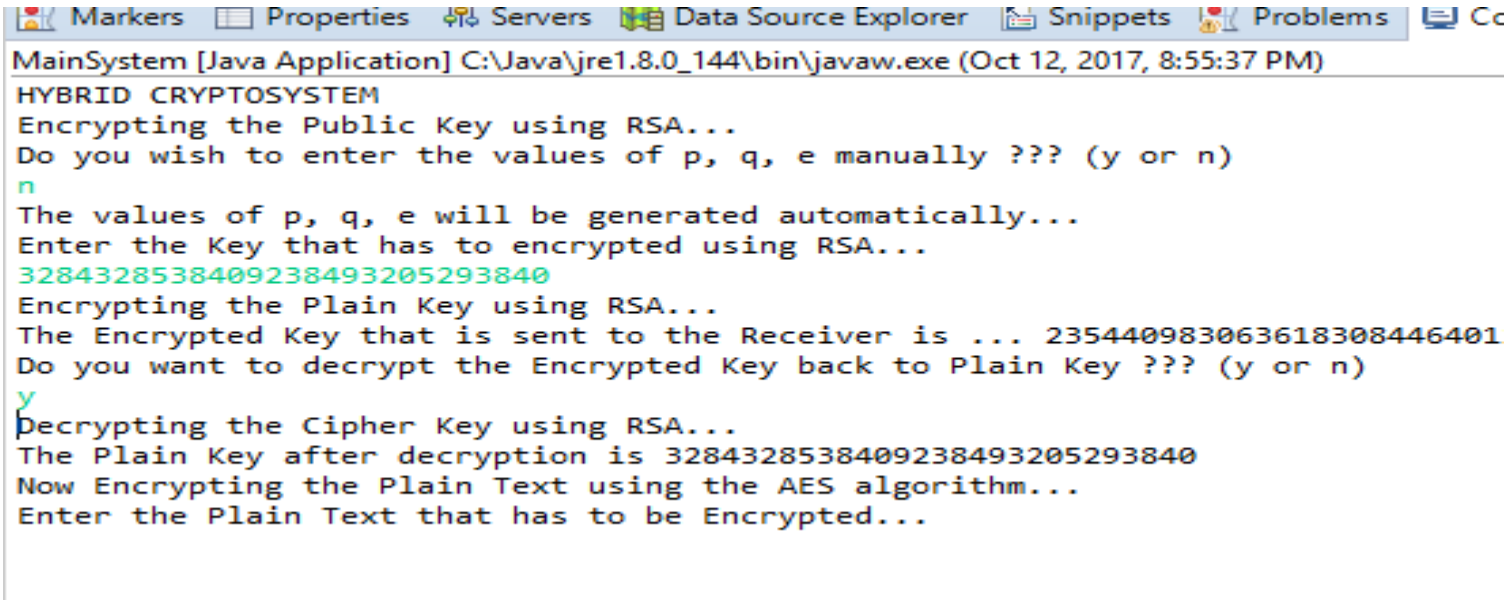


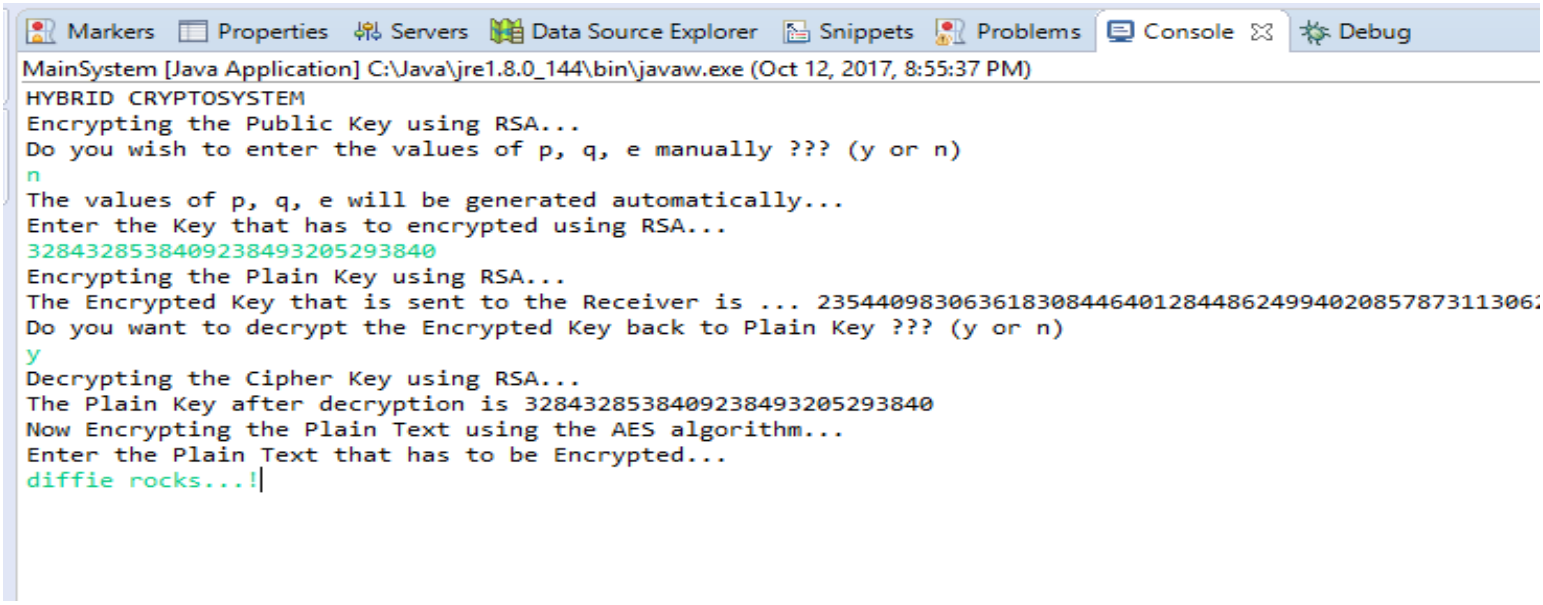


The user has to enter the plaintext message (the AES key).



After the user enters the message, it encrypts it.

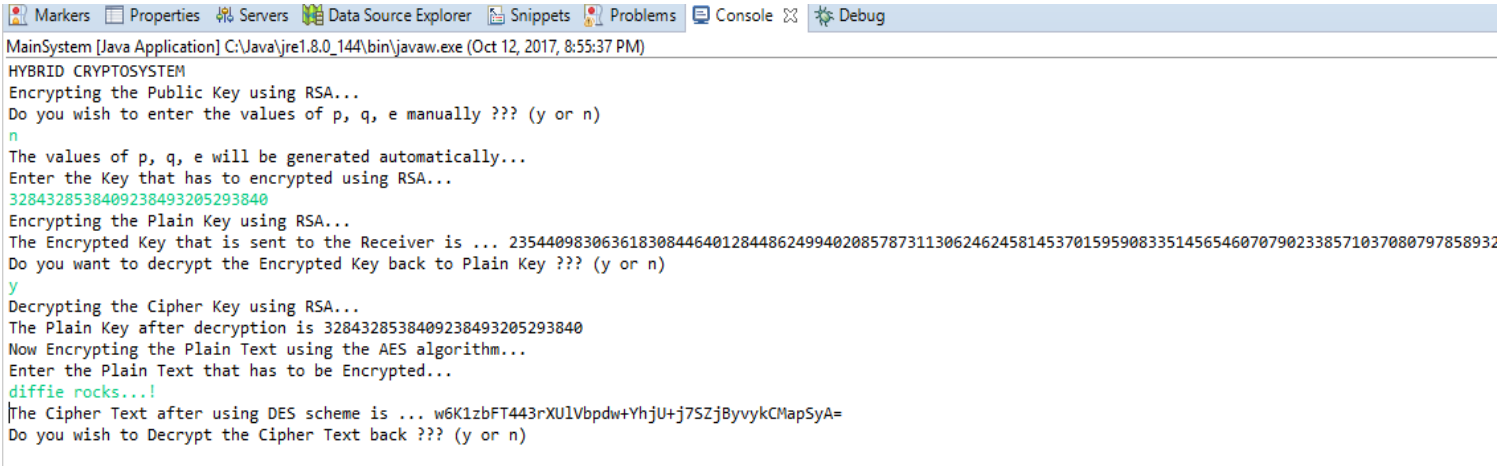




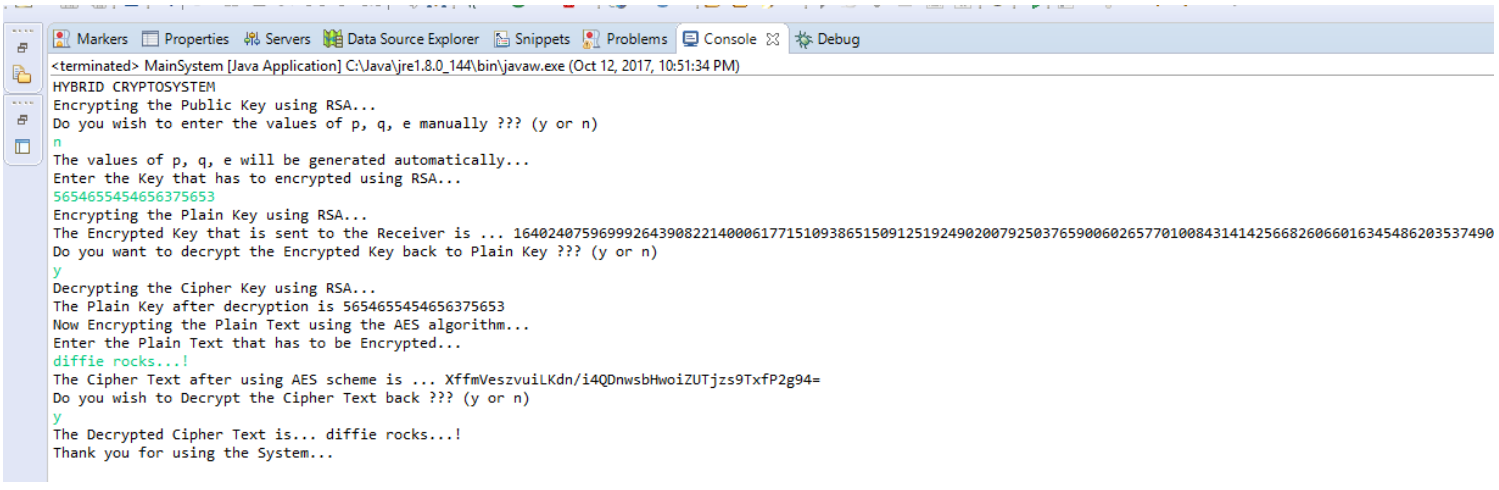
The user then enters the plaintext ("diffie rocks…!") that has to be encrypted to AES.

**To show different results, we are taking this plaintext instead of "Diffie Rocks !!!" (the solution to our challenge).*

You will see different encrypted key being generated in "Part II", Crypto Attack. (Page 13).



The user gets the corresponding ciphertext.



Part II: Crypto Attack (mysterytwisterc3.org challenge)

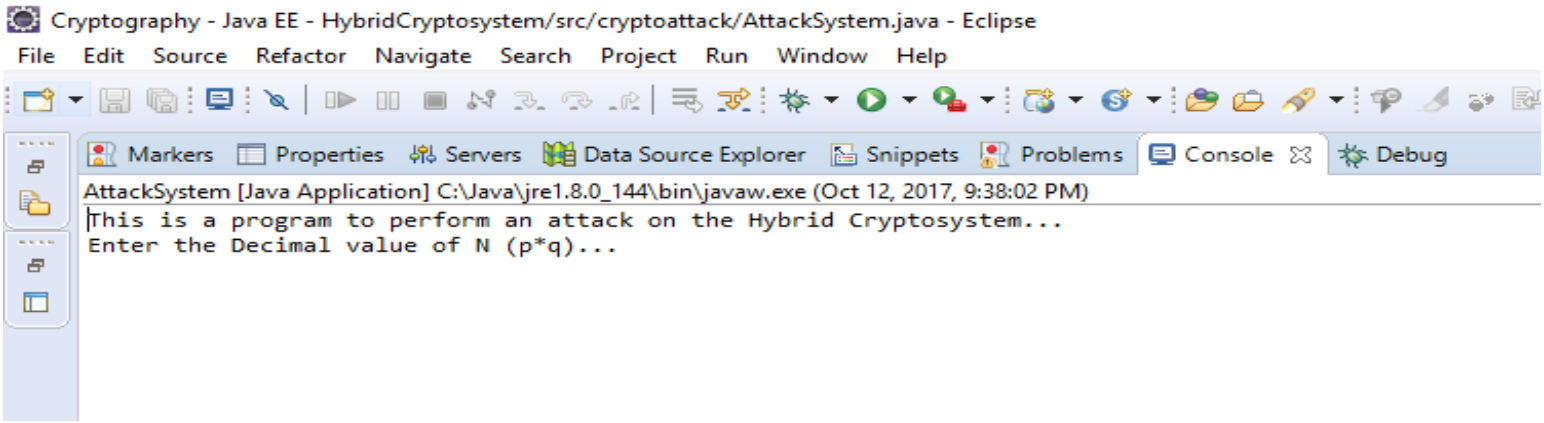
Details given:

C_s = 0xfd0b934c23288975648cd1d03ed3c5e2

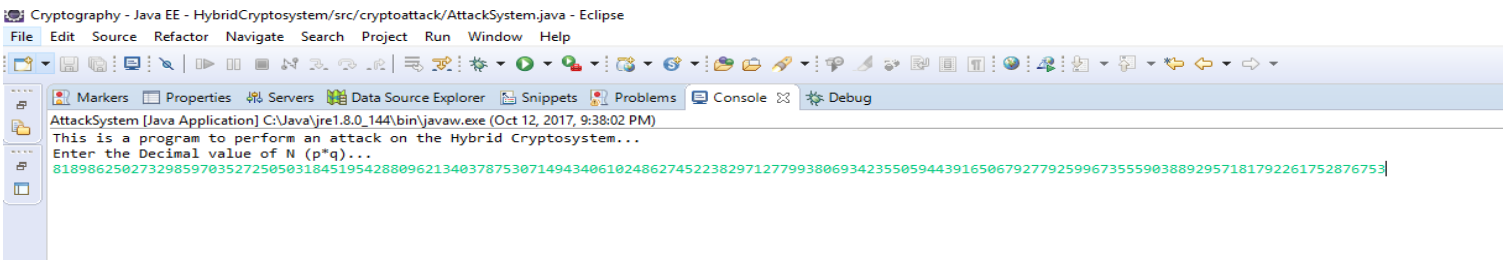
C_p = 0xc0eacf32dc0492464d9616fefc3d01f56589a137781bf6cf56784dea1c44ef52d61b1025655f370eb78646716f93e0a5

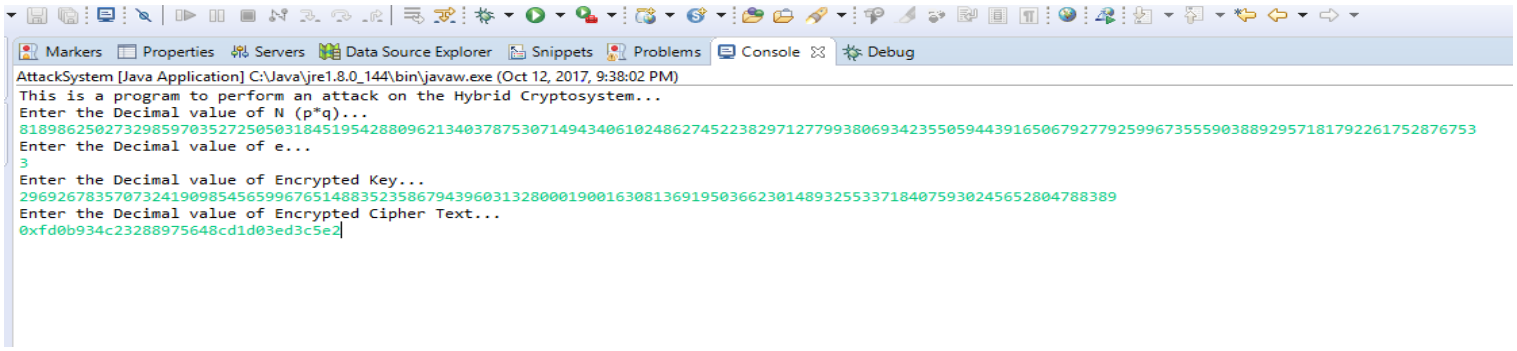
n = 0x9c5f36caf9adc60b4447897c639f1564ed0709251147276de030db395555c8afed912a198b334bd2301981731282981 26e958e38cac653e061035e300505eed1

e = 0x3

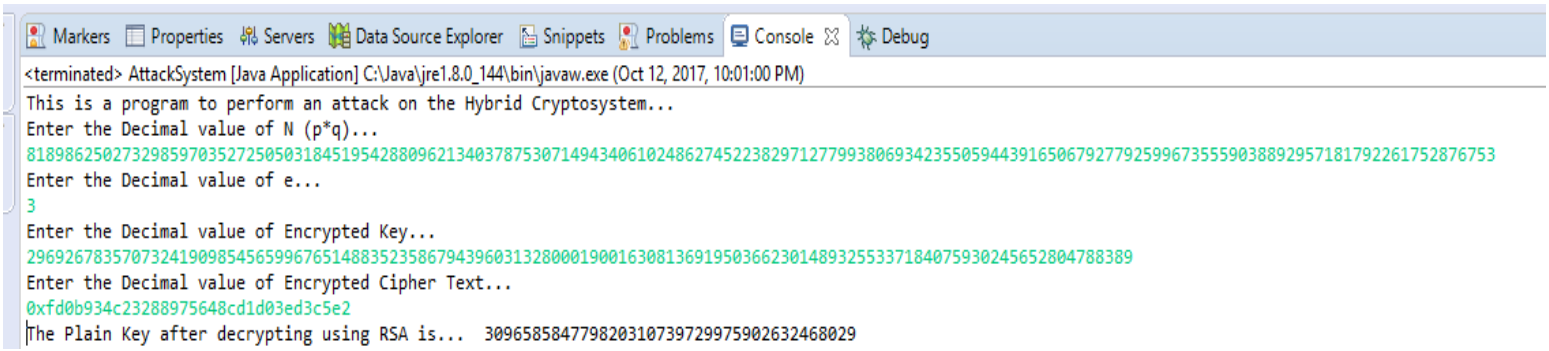


The user enters the decimal value of N.

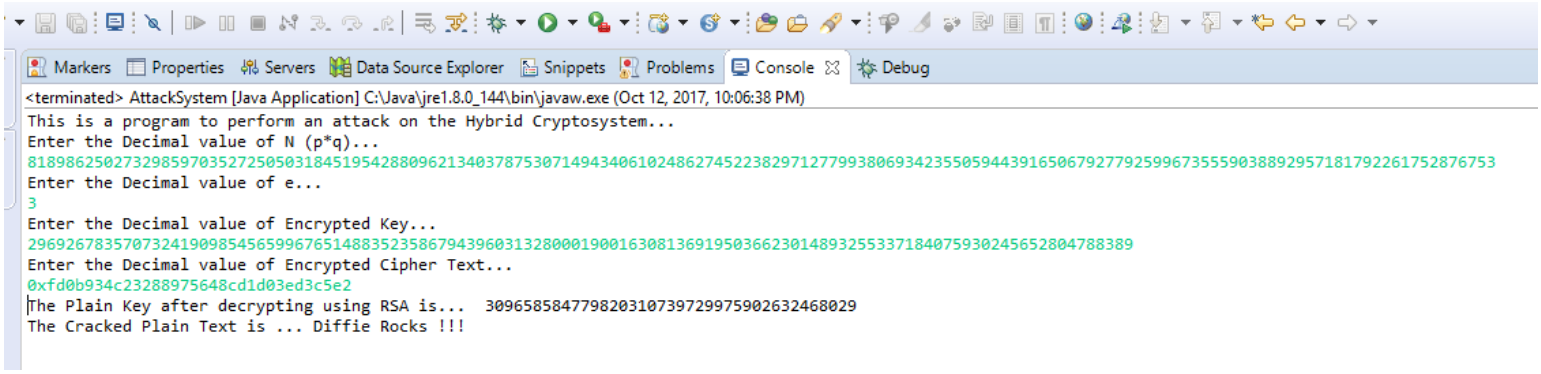




Then the user has to enter the values of e, K and c_s.



After this, the user gets the decrypted key K.



This key K is used to decrypt the AES plaintext message, which is “Diffie Rocks !!!”

**Note that this is different from the values entered in Part I, Page 11.*

Proof of successful challenge completion (Hybrid Encryption I challenged solved/cracked)