ECE 40400: Introduction to Computer Security

Spring 2024

Prateek Yashwant Jannu

Purdue University, West Lafayette

Professor: Dr. Avinash C Kak

Problem 1: RSA encryption and decryption

Ciphertext:

```
5794d0da2de74c58cae4959f4df22c3f824c45696c21707c5a03645e2e5b1c4dc2285609083d80354
a2befced1edca573115bcdd5ab634295d46645d19c347213a10441f1b2196dfae8c88f13a873cbb56f
5debf2a64a102e8a6fc908991a11e8a52e8b8197581aa4cc8dbab2c987659498b2c0cb39085b0ce57
9d91666966349428e337e8a7d63ea27abbf75b85347c025fe39e8ab2422a770c48900210748cbaf40
6182119eed41fd36c2fb266aa0f32b946b1c2c47783915f28be62659d2f635311ec8f1062d27cd94c
460c1d964c8f94257aa8d5f232442bed6f349603c43dc32eb3d6101f3a1974a8bc4b42d14ecc8c8f5
1a5ed7d69c406ccfaa9a808cf35ac5cb76bd9bf94c7b72ff7964b7a05c0170f6132c19e02088c01b5c
450b5ba13b960f8eaaf727b5e16fe6e67cd89f3974449592e7b2129ca7db380fbeadc497c8e4ab80be
ddad0ec54face12781e4fd6d41f8b297a117083e0ab11bea6088528edbae36020332cc86913c6fb18
80bef47a941d8f98b39192533f3883d73d644f9823a0d0dbed50641666804848b43427a7326f0934
d311df014cb5102f58da30798fc4e8cd8c096b4f463e124c5898c04b00bd80ba2ebd972ab96c727f8
a1498337d10515684587ae45d836113f69ba0e32528e6b25ede4e9712d61ee5ad9b698020255b7b5
7005d5c8f65337abdf19bb8ac1776a4bd3b9336733c9174737c5ac12050c4b3efe5607d4a36bb2cd9
c90c7a31ac6ff4bd2b0d9d40314b9ff09f9f8196f1600d1f32a13b6941e108f38f97f4583fb28531eecf
a46b5265a8b260a072af5590b5a397afedd530db823855da5581940d444a9b3f9e1354610c085363
2ea94a58ac4285dcd892f05ca922b1376e1333e56dc16b91398f5cd376d056881496ce8f812c5eb38
8ba7b37402a257ee5b3a343b7591e21dbdca35460c93294e5bfdf33ddd379702f00b75f9d686f556f
e233e222aabf076e745e089aa499058a2371d697170a7bc359b82e4cd85c41c4a4cb0970302eb966
909e0f4492865d0d935e4f4ab4e99ea833cc2c710eaaf6491559e292db63b28981fe37397db813ea9f
1d9d7b4da91c475ed78e5419799d7c5df44be35ff49751796130f98b355cba96666c11fa360e5a372
a30d3f5c8ae0f2bd575443d8975543aaaaaea6bb03a8985177f9487f382c9ea76529154ee80f88e6d2
11be998ab2918b2efbb629243110487e2d9c895e36d076b586ecd3c961592e007984b2294dbb8ade
05d5f4fab75f906d9771a3d5bc8a25218be02eb2259f81cf996ce6cb65dfa7d09d6461e9ebc5f51670
164ca5d167fd1a785cf2847726dbd6fced35fca4ddb686f19ddb6290e4f010bf6f1c2f0194c59a4c247
d9fc182b67a820fe0cad02ff9db3e45eee54c67dac2791099b429af5b4ef43f24bd771f3c364b1fabb8
d146f95c90dd16e0f6ec44f1281cdf46cf63a92ff6e8f733d37bc292e3489826a26448d32b174b3020
e913466562b7f875f757ff03d1915d6036d356123491b1be7cc57f6c261ff65dbf797cf8616e302018
bccc81777a6fac6402e10ac5ce404bd796887ae840e3f49a1643fb9b8173e6ea92d1421bea0f6da030
25b6dee1a2475ca877fd24ede647cacce93df5ae5c18204262639c2c9e4da59f723b1a84dfb9e8fc54
e7188163528b580c0c40ecd767cf9051ed1adb283fb37e4500412d6d82e479290c3321d05d653572
5d4b19be95421fb8cf0661bff980b860243b95ed13e59e602473f1f
```

Below is a detailed explanation of how it works:

Init Method:

1. If the script is invoked with either -e or -d command line arguments, it reads two files containing prime numbers (p and q) and calculates the RSA parameters: modulus (n), public exponent (e), and private exponent (d). It also computes Euler's totient function (totient) which is used in the RSA key generation process.

2. If invoked with -g command line argument, it initiates the object with the provided public exponent (e).

Encrypt Method:

1. Reads plaintext from the specified file (plaintext) and converts it into a BitVector.
2. Breaks the plaintext into 128-bit blocks.
3. For each block, pad it to 256 bits, raise it to the power of e, and takes the modulus n.
4. Write the encrypted blocks as hexadecimal strings to the specified ciphertext file.

Generate Method:
1. Generates two random prime numbers (p and q) of 128 bits using the PrimeGenerator class.
2. Check if conditions for RSA keys are satisfied: p and q are distinct, both start with the two most significant bits set to 1, and Euler's totient function of p and q are coprime with e.
3. Write p and q to the specified files.

Decrypt Method:
1. Reads ciphertext from the specified file (ciphertext) and converts it into a BitVector.
2. Breaks the ciphertext into 256-bit blocks.
3. For each block, calculate V_of_p and V_of_q using the private exponent d and the Chinese Remainder Theorem.
4. Combines V_of_p and V_of_q using the Chinese Remainder Theorem.
5. Write the resulting blocks as ASCII characters to the specified recovered plaintext file.

Main(Given in question)
1. Initiates an instance of the RSA class with a public exponent of 65537.
2. Based on the command line argument (sys.argv[1]), either encrypts, decrypts, or generates keys.

Problem 2: Breaking RSA Using CRT

Explanation:

Break_rsa Method:
1. Reads three ciphertext files (enc1, enc2, enc3) containing RSA-encrypted messages, along with a file (n_1_2_3.txt) containing the modulus values (n1, n2, n3) used for encryption.
2. Parses the ciphertexts and modulus values.
3. Calculates the combined modulus (n_combined = n1 * n2 * n3).
4. Computes the inverse of each n_i modulo n_combined. These inverses (CRT_step_two) are used later in the CRT calculation.
5. Iterates over the ciphertexts, processing 256-bit blocks of each ciphertext.
6. For each block, applies the CRT by calculating (M^3) using the Chinese Remainder Theorem formula: (M^3 = sum{i=1} to {3} (C_i * M_i *times X_i) mod n), where (C_i) is the ciphertext, (M_i) is the product of all moduli except (n_i), and (X_i) is the modular multiplicative inverse of (M_i) modulo (n_i).

7. Calculate the cube root of M^3 using the solve_pRoot function to obtain the decrypted message (M).
8. Write the decrypted message to the output file (cracked.txt).

Generate Function:
This function generates random prime numbers p and q and calculates their product n. It ensures that both p and q are distinct, have their two most significant bits set to 1, and are coprime with the given public exponent e=3.

Main Block:
1. Creates an instance of the break RSA class with a public exponent e=3.
2. If invoked with -e command line argument, generates three sets of RSA keys (n, p, q) using the generate function, and encrypts the plaintext using each set of keys. Write the modulus n values to the output file (n123.txt).
3. If invoked with -c command line argument, decrypts the three ciphertexts using the CRT method and writes the decrypted message to the output file.
4. If invoked with -g command line argument, generates RSA keys (p and q) and writes them to the specified files (p_text and q_text).

Running the Script:
1. To encrypt a message, run the script with -e option followed by the message file name, and three ciphertext file names.
2. To break RSA encryption, run the script with -c option followed by three ciphertext file names, n_1_2_3.txt, and the output file name.