

Encrypting and Decrypting with RSA

In this exercise, you will use a simplified version of the RSA algorithm to encrypt and decrypt numbers. You should first imagine yourself as Bob and complete his tasks, then pass some information (but don't pass the entire exercise sheet!) to Alice (still in your own role). Afterward, complete the exercise as Alice. Try to complete all steps manually as much as possible. For larger calculations, you can write a short loop. You can do this in the Python interactive prompt or in the source file.

Bob will generate an RSA key and share his key with Alice. Alice will send Bob message. Trudy will try to read Alice's message.

Instructions for Bob:

We will be doing $B=16$ -bit RSA. Select the encryption exponent $e=17$. (In practice, $e=65537$ would often be used for larger p and q .)

1. Calculate p like this: (Write results in the table below)
 - a. Create an 8-bit binary number by flipping a coin (or using some other random number generator) 5 times to select 5 random bits, and then append 11____1 around your number. This forms our tentative p .
[To create the same sort of number in Python, first `import random`. Next, select an $(B/2=)$ 8-bit random number using `random.randint(i, j)` to select an integer x satisfying $i \leq x \leq j$. Use `bits()` to view the binary form. Then, set the two highest bits and the lowest bit (to 1) using $p = p | 0b11000001$. (`|` is bitwise or)]
 - b. Check if p is prime. If p is not prime, add 2 to p and try again. (You can check that a number is prime by hand, or use a program, e.g., check if all numbers smaller than p do not divide p . Avoid googling.).
 - c. Check if the number $(p-1)$ is co-prime with $e=17$, i.e., $\gcd(p-1, e)=1$. If not, add 2 to p and try again. This step is necessary to ensure that we can find a d such that $ed = 1 \pmod{z}$
Note: since $e=17$ is prime, you can simply check that $(p-1) \bmod e \neq 0$.

Initial random number (decimal)	Initial random number (binary)	p (decimal)	p (binary)	Is p prime?	Is $(p-1) \% e \neq 0$?
245	11110101	245	11110101	no	no
		247	11110111	no	no
		249	11111001	no	no
		251	11111011	yes	yes
		Final: 251	11111011	yes	yes

(Instructions for Bob, continued.)

2. Repeat step 2 to select q . (Note: q must be different from p . Start over if q will equal p .)

Initial random number (decimal)	Initial random number (binary)	q (decimal)	q (binary)	Is q prime?	Is
---------------------------------	--------------------------------	---------------	--------------	---------------	----

					$(q-1) \% e \neq 0$?
221	11011101	221	11011101	no	no
		223	11011110	yes	yes
		Final: 223	11011110	yes	yes

3. Calculate the modulus $n = pq$

$$n = 55973$$

4. Calculate the totient $z = (p-1) * (q-1)$

$$z = 55500$$

5. Select the decryption exponent d such that $(e * d) \bmod z = 1$. You can “guess and check” all values of $1 < d < z$. Only one value of d will work if z is selected as in step 5, and p and q are selected as in steps 2 and 3. This is your private key. Do not reveal d , p , q , or z to Alice or Trudy!

$$d = 22853$$

6. Provide your public key (e, n) , in other words the numbers e and n , to Alice and Trudy. This simulates posting your public key for the world to see.
7. Wait for Alice to send you a secret message. (See the next page.)
8. Once you receive the secret message from Alice, you can decrypt it using your private key. Suppose c is the ciphertext. Compute the original message m as $m = c^d \bmod n$.

$$c = 1185$$

$$m = 21$$

Don't reveal the secret message to Trudy!

Instructions for Alice:

1. You will receive the public key (e, n) , in other words the numbers e and n , from Bob. Write it here:

$e = 17$

$n = 55973$

2. Select any number $0 \leq m < n$ for your plaintext secret message. If you like, you can encrypt a sequence of ASCII characters as separate messages m (that is, using block encryption.)

$m = 21$

3. Compute the ciphertext c as $c = m^e \bmod n$.

$c = 1185$

4. Give the ciphertext message c to Bob and Trudy. This simulates Trudy eavesdropping on the wire.

Additional Tips

To computing $m^e \bmod n$ and $c^d \bmod n$ can overflow your calculator especially for large values of d . There is a trick you can use that takes advantage of the fact that the modulus is taken of the result of the exponentiation. You can find out more here:

https://en.wikipedia.org/wiki/Modular_exponentiation

This pseudocode might be helpful:

```
function modular_pow(base, exponent, modulus) is
    result := 1
    base := base mod modulus
    while exponent > 0 do
        if (exponent mod 2 == 1) then
            result := (result * base) mod modulus
        exponent := exponent >> 1
        base := (base * base) mod modulus
    return result
```

Instructions for Trudy: (optional, 5 bonus points)

(If you have extra time, you may want to play this role)

1. Wait to receive the public key (e, n) , in other words the numbers e and n from Bob.

$$e = 17$$

$$n = 73903$$

2. Factor n to find p and q . Use a brute-force Python loop. This is the hard step that makes RSA secure for large numbers.

$$p = 263$$

$$q = 281$$

3. Compute $z = (p-1) * (q-1)$

$$z = 73,360$$

4. Now compute d the same way as Bob did: Select the decryption exponent d such that $(e * d) \bmod z = 1$. You can simply “guess and check” all values of $d < z$. Only one value of d will work for a given e and z .

$$d = 43,153$$

5. Wait to receive (eavesdrop) on the ciphertext message c from Alice to Bob.

$$c = 14,379$$

6. Decrypt the c , ciphertext message: Compute the original message m as $m = c^d \bmod n$.

$$m = 12,345$$

Acknowledgement: The simple form of the RSA encryption/decryption used in this exercise is based on Avi Kak's lecture notes on cryptography, available at

<https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture12.pdf>

and from the text, Kurose & Ross, Computer Networking: A Top-Down Approach, 7th Edition, Section 8.2.2, pp. 606-610 (6th ed: same section, pp. 684-688).