# 3. Public-key Cryptography - RSA

## 3.1 OVERVIEW

### 3.1.1 Introduction and learning objective

Symmetric encryption has evolved from classical to modern but also leaves a lot to be desire. Two of the most difficult problems associated with symmetric encryption are:

- ***Key distribution*** *- the issue of secret key exchange between sender and receiver*: Key distribution under symmetric encryption requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2) the use of a key distribution center. A secure channel is required for key exchange so that the key must be kept secret and known only to the sender and receiver. This will be difficult to implement and establishing such a secure channel will be costly and time-consuming.
- ***Digital signatures*** *and confidentiality of keys and* : If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents. Besides, there is no basis to blame if the key is revealed.

Diffie and Hellman achieved an astounding breakthrough in 1976, by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography, going back over four millennia. That is *public key cryptography* or *asymmetric cryptography*. The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography.

To discriminate between the two, we refer to the key used in symmetric encryption as a **secret key**. The two keys used for asymmetric encryption are referred to as the **public key** (are often denoted as *PU*) and the **private key** (are often denoted as *PR*). The plaintext is denoted by M, the ciphertext is denoted by C

There are 2 main approaches in Public-key cryptography:

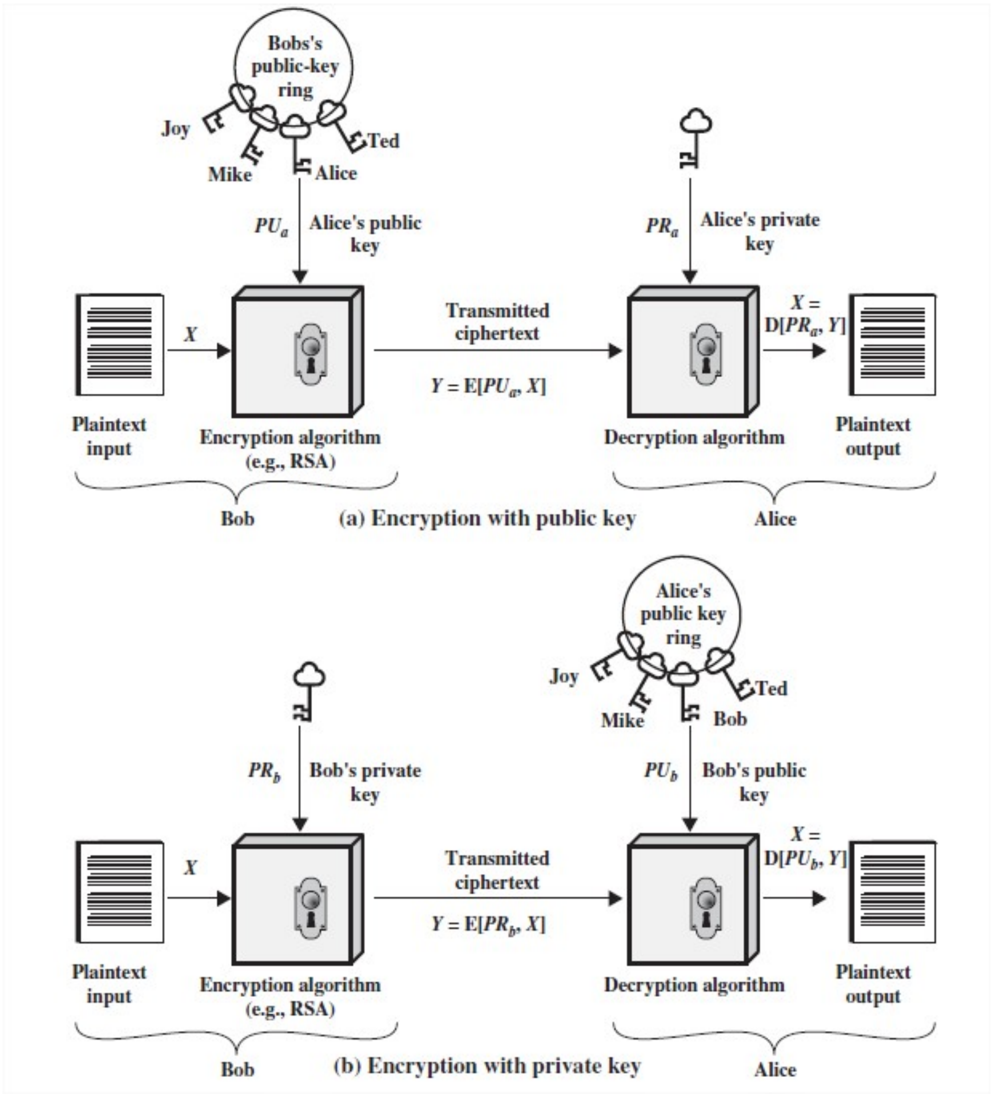

Figure 3.1: Two approaches in Public-Key Cryptography

1. **Public-Key Cryptosystem for Confidentiality**:
   Encryption with public key (Figure 3.1). If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.
   - Encryption: **C = E(M,** ***PU*****)**
   - Decryption: **M = D(C,** ***PR*****)**
2. **Public-Key Cryptosystem for Authentication**:
   Encryption with public key (Figure 3.1). In this case, Alice prepares a message to Bob and encrypts it using Alice's private key before transmitting it. Bob can decrypt the message using Alice's public key. Because the message was encrypted using Alice's private key, only Alice could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature**.

- Encryption: **C = E(M,** ***PR*****)**
- Decryption: **M = D(C,** ***PU*****)**

The *learning objective* of this lab is for students to gain hands-on experiences on the RSA algorithm. From lectures, students should have learned the theoretic part of the RSA algorithm, so they know mathematically how to generate public/private keys and how to perform encryption/decryption and signature generation/verification. This lab enhances student's understanding of RSA by requiring them to go through every essential step of the RSA algorithm on actual numbers, so they can apply the theories learned from the class. Essentially, students will be implementing the RSA algorithm using a program language.

### 3.1.2 Background

To complete this lab well, you are expected to gain knowledge about:
- Number Theory:
  + Modular Arithmetic
  + Divisor, the greatest common divisor *(Euclidean Algorithm)*
  + Prime numbers, Testing for Primality
- RSA Cipher

If you are not familiar with them, you should find out in more detail in:
**Chapter 2: Introduction to Number Theory & Chapter 9: Public-Key Cryptography and RSA**
W. Stallings, *CS book - Cryptography and network security: Principles and practice, 7th ed*. Boston, MA, United States: Prentice Hall, 2017

### 3.1.3 Lab environment and Tools

1. **Operating system:** Windows, Linux (Ubuntu), MacOS
2. **Programing languages and IDE:** Flexible, you are free to choose any programming language you wish (Python, Golang are highly recommended)
3. **Tools:** CrypTool 2 - `https://www.cryptool.org/`.

## 3.2 LAB TASKS

### 3.2.1 Number Theory

Before starting with the public key encryption algorithm, we will kick-off with some number theory revision, specifically check for prime numbers, greatest common divisor (GCD), modulo.

**Task 3.1** Write a program (with your own programming language) to fulfill the following requirements:
1. Prime number:
   + Generate a random prime numbers with 2 bytes, 8 bytes, 32 bytes long
   + Determine 10 largest prime number under 10 first **Mersenne** prime numbers.
   + Check if an arbitrary integer less than $2^{89}-1$ is prime or not
2. Determine the greatest common divisor (gcd) of 2 arbitrary "large" integers (which are as large as possible that you can handle)
3. Compute the modular exponentiation $a^x \, mod \, p$. Your program should be able to compute in case of "large" exponents (x>40), for example $7^{40} mod \, 19$

■

**Tips 3.1.**
*- To check large prime numbers, you can find out and use the Miller-Rabin[1] algorithm. You can find out about Mersenne prime number at `https://en.wikipedia.org/wiki/Mersenne_prime`.*
*- To determine the greatest common divisor, you should use Euclid algorithm*
*- Modular multiplication has the following property:*
$(a \times b) \equiv [(a mod\, n) \times (b mod\, n)] mod\, n$
*It can be applied in compute modular exponentiation in case of "large" exponents.*

## 3.2.2 RSA Public-Key Encryption

RSA is one of the first public-key cryptosystems and is widely used for secure communication. This cipher was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. In RSA scheme, the plaintext and ciphertext are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than $2^{1024}$. RSA algorithm can be summarized as follows.

1. Select two "large" prime numbers p and q ($p \neq q$), then calculate n = pq
2. Calculate $\phi(n) = (p-1)(q-1)$
3. Select e such that e is relatively prime to $\phi(n)$ and less than $\phi(n)$.
4. Determine d such that $e.d \equiv 1 \, mod \, \phi(n)$ *(d can be calculated using the extended Euclid's algorithm)*
5. The resulting keys are public key $PU = (e,n)$ and private key $PR = (d,n)$
6. To encrypt a plaintext input of M:
   + Encryption for Confidentiality: $C = E(M,PU) = M^e mod\, n$
   + Encryption for Authentication: $C = E(M,PR) = M^d mod\, n$
7. To decrypt ciphertext input of C:
   + Decryption for Confidentiality: $M = D(C,PR) = C^d mod\, n$
   + Decryption for Authentication: $M = D(C,PU) = C^e mod\, n$

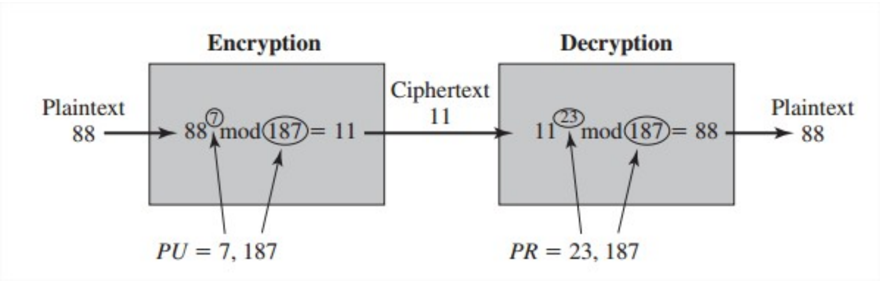

Figure 3.2: Example of RSA Algorithm

When using RSA to process multiple blocks of data, each plaintext symbol could be assigned a unique code of two decimal digits (e.g., a = 00, A = 26). A plaintext block consists of four decimal digits, or two alphanumeric characters. Figure 3.3.a illustrates the sequence of events for the encryption of multiple blocks, and Figure 3.3.b gives a specific example. The circled numbers indicate the order in which operations are performed.

[1] `https://asecuritysite.com/encryption/rabin`

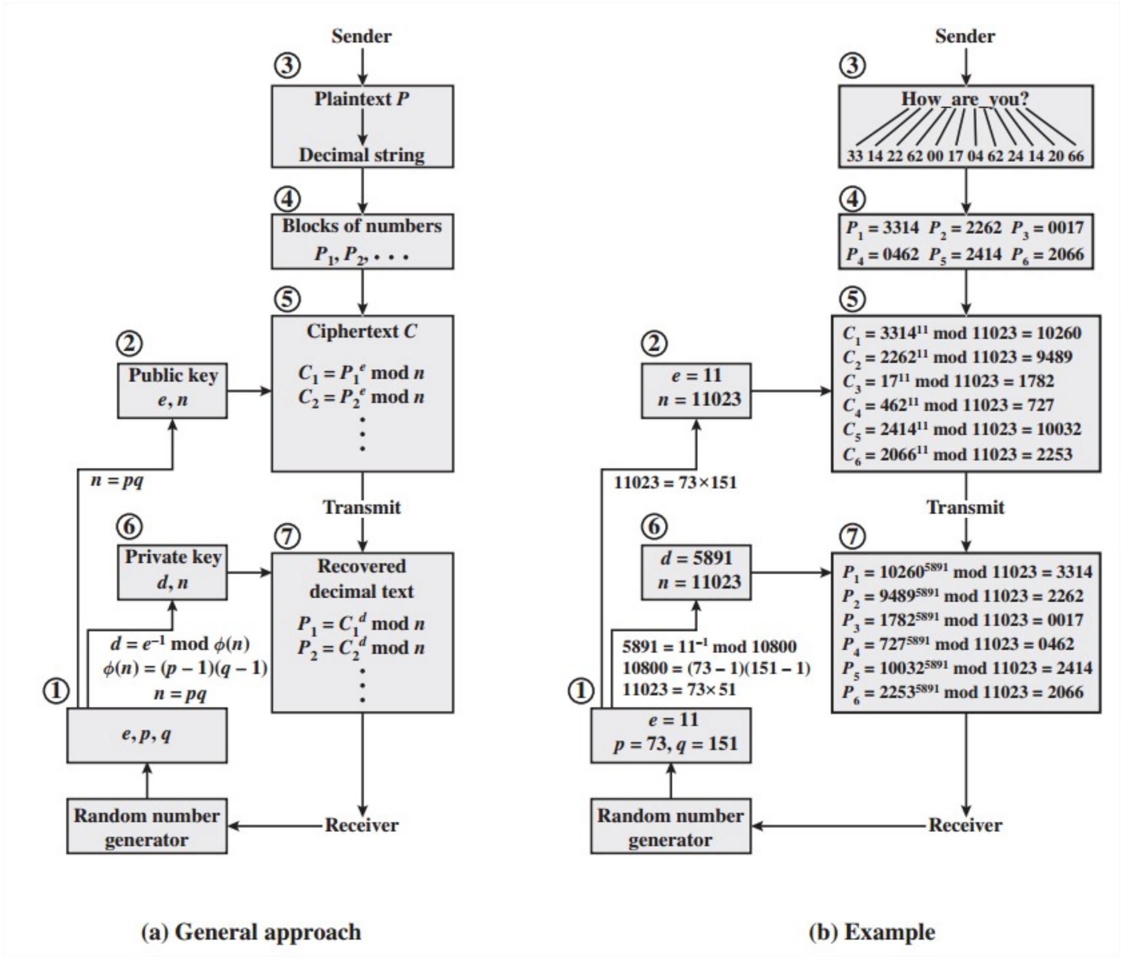
Sender
Plaintext P
Decimal string
Blocks of numbers
$P_1, P_2, \ldots$
Ciphertext C
$C_1 = P_1^e \bmod n$
$C_2 = P_2^e \bmod n$
Public key
e, n
n = pq
Transmit
Private key
d, n
Recovered decimal text
$P_1 = C_1^d \bmod n$
$P_2 = C_2^d \bmod n$
$d = e^{-1} \bmod \phi(n)$
$\phi(n) = (p-1)(q-1)$
n = pq
e, p, q
Random number generator
Receiver
(a) General approach
How_are_you?
33 14 22 62 00 17 04 62 24 14 20 66
$P_1 = 3314$ $P_2 = 2262$ $P_3 = 0017$
$P_4 = 0462$ $P_5 = 2414$ $P_6 = 2066$
$C_1 = 3314^{11} \bmod 11023 = 10260$
$C_2 = 2262^{11} \bmod 11023 = 9489$
$C_3 = 17^{11} \bmod 11023 = 1782$
$C_4 = 462^{11} \bmod 11023 = 727$
$C_5 = 2414^{11} \bmod 11023 = 10032$
$C_6 = 2066^{11} \bmod 11023 = 2253$
e = 11
n = 11023
11023 = 73×151
d = 5891
n = 11023
$P_1 = 10260^{5891} \bmod 11023 = 3314$
$P_2 = 9489^{5891} \bmod 11023 = 2262$
$P_3 = 1782^{5891} \bmod 11023 = 0017$
$P_4 = 727^{5891} \bmod 11023 = 0462$
$P_5 = 10032^{5891} \bmod 11023 = 2414$
$P_6 = 2253^{5891} \bmod 11023 = 2066$
$5891 = 11^{-1} \bmod 10800$
10800 = (73 – 1)(151 – 1)
11023 = 73×51
e = 11
p = 73, q = 151
(b) Example

Figure 3.3: RSA Processing of Multiple Blocks

**Task 3.2** To get acquainted with RSA, your task is using CrypTool 2 or other tools to perform the following experiments:

1. Determine public key *PU* and private key *PR*, if:
   + $p_1 = 11, q_1 = 17, e_1 = 7$ (decimal)
   + $p_2 = 20079993872842322116151219$,
   $q_2 = 676717145751736242170789, e_2 = 17$ (decimal)
   + $p_3 = F7E75FDC469067FFDC4E847C51F452DF$,
   $q_3 = E85CED54AF57E53E092113E62F436F4F, e_3 = 0D88C3$ (hexadecimal)
   *(Note: Remember to check if the above values are prime numbers or not before calculating the keys)*
2. Using key which generated by $p_1, q_1, e_1$ to encrypt and decrypt the plaintext M=5 in both case *Encryption for Confidentiality* and *Encryption for Authentication*.
3. Use the keys above to encrypt the following message:
   **The University of Information Technology**
   Determine the ciphertext as Base64.
4. Find the corresponding plain-text of each following ciphertext, knowing that they are encrypted by one of the three given keys above.

(a) raUcesUlOkx/8ZhgodMoo0Uu18sC20yXlQFevSu7W/
FDxIy0YRHMyXcHdD9PBvIT2aUft5fCQEGomiVVPv4I
(b) C8 7F 57 0F C4 F6 99 CE C2 40 20 C6 F5 42 21 AB AB 2C E0 C3
(c) Z2BUSkJcg0w4XEpgm0JcMExEQmBlVH6dYEp
NTHpMHptMQ7NgTHlgQrNMQ2BKTQ==
(d) 00101000 00010100 11111111 10110111 00101110 11001010
11101100 01100111 10111111 00111111 01101000 11001111 00110000
10010100 01010100 11110101 01001100 11101110 11101111 01011011
00000100

**Tips 3.2.** *You can use the RSA Cipher template or build their own template to solve these problem in CrypTool 2*

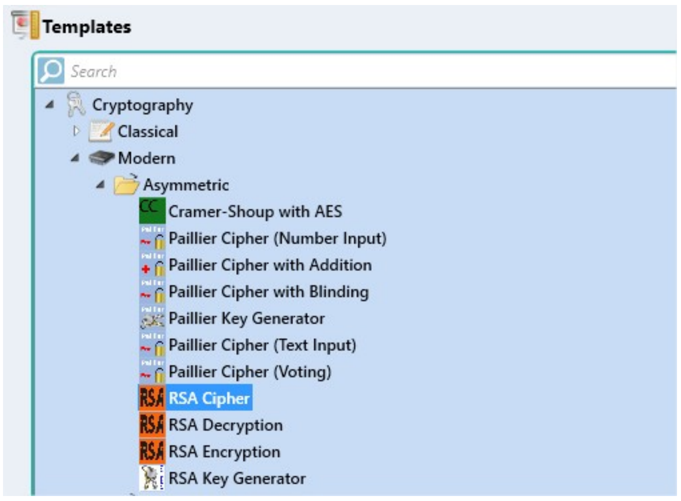

Figure 3.4: RSA Cipher template in CrypTool 2

**Task 3.3** Your task is to write a program to illustrates how simple RSA cipher work, meet the following requirements:

- Generating keypair (PU, PR) using the given "valid" inputs *p, q, e* or generate a randomized keypair if *p, q, e* are not given.
  *Note that keypair is as "large" as possible*
- Use the generated keys to encrypt/decrypt the message. The message can be numeric or string.

Check the results of your application with some example input as at **Task 3.2**.

**Tips 3.3.** *You need to solve some problems when writing this applications such as: Generating/test large prime numbers (can use Miller-Rabin algorithm); find the greatest common divisor (can use Euclid's algorithm); applying modular multiplication and exponentiation properties to calculate "large" $a^x$ modn, ...*

**Advanced Task 3.1** The alternative tasks for this lab that you can consider to conduct are all tasks in **RSA Public-Key Encryption and Signature Lab** - SEED Labs. You can find out the full version of this SEED lab here: `https://seedsecuritylabs.org/Labs_16.04/Crypto/Crypto_RSA/`.

## 3.3 REQUIREMENTS - EVALUATION

### 3.3.1 Requirements

You are expected to complete all tasks in section **Lab Tasks**, advanced tasks are optional and you could get bonus points for completing those tasks. You can either practice individually or work in team (2 members/team). If you prefer to work in team, please register in the first class with instructor and keep working in your team afterwards.

Your submission must meet the following requirements:

- You need to submit a **detailed lab report in .PDF** format, **using report template** that was provided on the courses website. You need to provide screenshots, to describe what you have done and what you have observed and explanation to the observations that are interesting or surprising.
- **Both of Vietnamese and English report are accepted**, that's up to you. Students in High Quality and Honor program are expected to write lab report in English.
- **Students in Honor program are expected to finish all advanced tasks.**
- When it comes to **programming tasks**, please attach all source-code and executable files (if any) in your submission. Please also list the important code snippets followed by explanation and screenshots when running your application. Simply attaching code without any explanation will not receive points.
- **Submit work you are proud of - don't be sloppy and lazy!**
  Your submissions must be your own work. You are free to discuss with other classmates to find the solution. However, report-copy is prohibited. Both of report owner and copier are received *a special gift - Zero point*. Please remember to clearly cite any source of material (website, book,...) that influences your solution.

**Notice 3.3.1** Combine your lab report and all related files into a single ZIP file (.zip), name it as follow:

**StudentID1_StudentID2_ReportLabX**

*For example: 19520123_19520234_ReportLab3.zip*. Maximum file size: 10MB.
If it is larger than 10 MB, then you should upload to Google Drive and submit the link to your report *(remember to share view permission with instructor)*

### 3.3.2 Evaluation

- Well complete all basic tasks: 70% *or 50% with students in Honor program (.ANTN)*
- Well complete the advanced tasks: 10-100% or bonus points to the next lab.
- In-class activities: + up to 10 points
- Work individually: **+ 2 point** *(updated)*
- Report written in English: + up to 2 points

**Notice 3.3.2** Assignments are expected to be completed by due date. For every day the assignment is late after the deadline, 10% will be deducted from the lab score. No assignments will be accepted once they are 7 or more days late.

Any part presented in your report may be randomly examine at the next class to verify your work. Absence without any rational reason could result in 30% deduction (or more) of your team's score.

## 3.4 REFERENCES

[1] William Stallings, *Cryptography and network security: Principles and practice, 7th ed*, Pearson Education, 2017. *Chapter 9: Public-key Cryptography and RSA*
[2] Wenliang Du (Syracuse University), *SEED Cryptography Labs*
`https://seedsecuritylabs.org/Labs_16.04/Crypto/`.
[3] Bernhard Esslinger et al., *The CrypTool Book: Learning and Experiencing Cryptography with CrypTool and SageMath, 12th ed, 2018*. Available: `https://www.cryptool.org/en/ctp-documentation`.

**Training platforms and related materials**

- ASecuritySite - `https://asecuritysite.com`
- Cryptopals - `https://cryptopals.com`

**Attention**: *Don't share any materials (slides, readings, assignments, labs,...) out of our class without my permission!*