

## Sécurité des Systèmes

# TP 2 : Signature digitale

**Filière :** Sécurité des Systèmes d'Information (SSI)

**Réalisé par les étudiantes :**

JOUIJATE Rim

AGOULZI Imane

**Professeur :**

AOUAD Siham

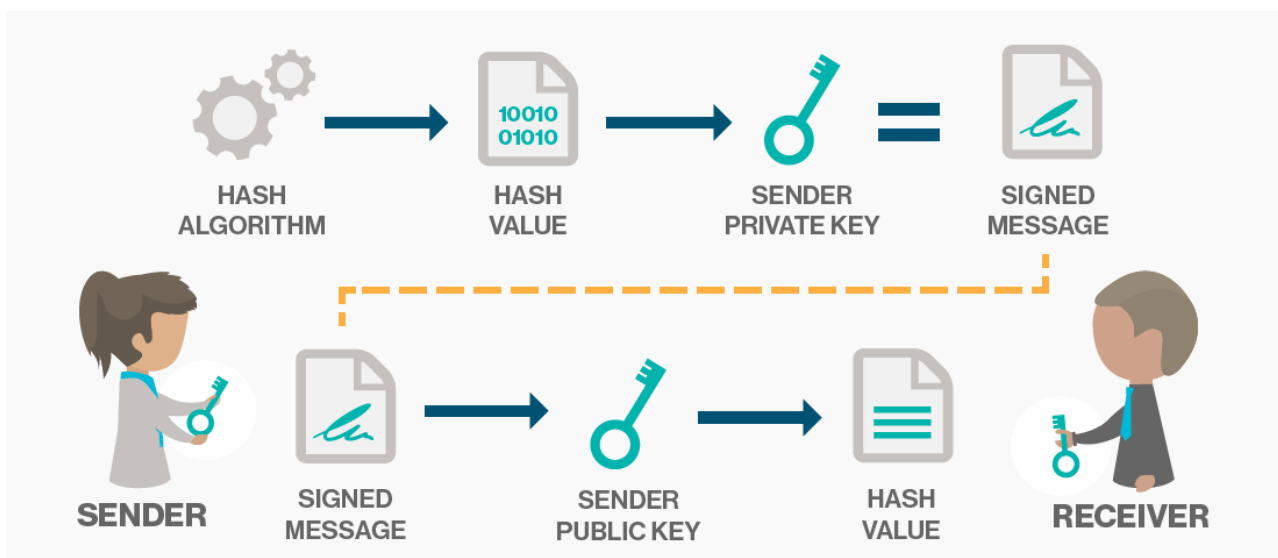
Dans ce TP, nous avons découvert les signatures numériques et comment sont-elles créées. Et ensuite nous avons créer une interface qui permet de signer une chaine de caractères avant de l'envoyer au destinataire en utilisant aussi l'algorithme de cryptage asymétrique RSA.

## 1 – Généralités sur les signatures digitales :

### Signature digitale :

Une signature : est une méthode utilisée pour garantir l'authenticité et l'intégrité d'un message, d'un document ou d'un ensemble de données numériques.

La signature numérique : est une forme de signature qui utilise des algorithmes cryptographiques, comme RSA ou DSA, pour créer une empreinte numérique unique du message envoyé. Cette empreinte numérique, appelée hachage, est ensuite cryptée avec une clé privée selon l'algorithme utilisé pour créer la signature numérique. La signature numérique est vérifiée en utilisant la clé publique associée à la clé privée utilisée pour signer le message. Si la signature est valide, cela prouve que le message n'a pas été modifié et que le signataire est bien celui qu'il prétend être.



### L'algorithme RSA :

L'algorithme RSA (Rivest-Shamir-Adleman) est un algorithme de cryptographie asymétrique qui est utilisé pour chiffrer et déchiffrer des données. Cet algorithme utilise deux clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement.

La clé publique est distribuée publiquement, tandis que la clé privée est gardée secrète.

Pour chiffrer un message, on utilise la clé publique du destinataire. Le message chiffré ne peut être déchiffré que par la personne possédant la clé privée correspondante.

Pour casse l'algorithme RSA, il nécessite la factorisation du nombre  $n$  en le produit initial des nombres  $p$  et  $q$ , avec les algorithmes classiques, le temps que prend cette factorisation croît exponentiellement avec la longueur de la clé.

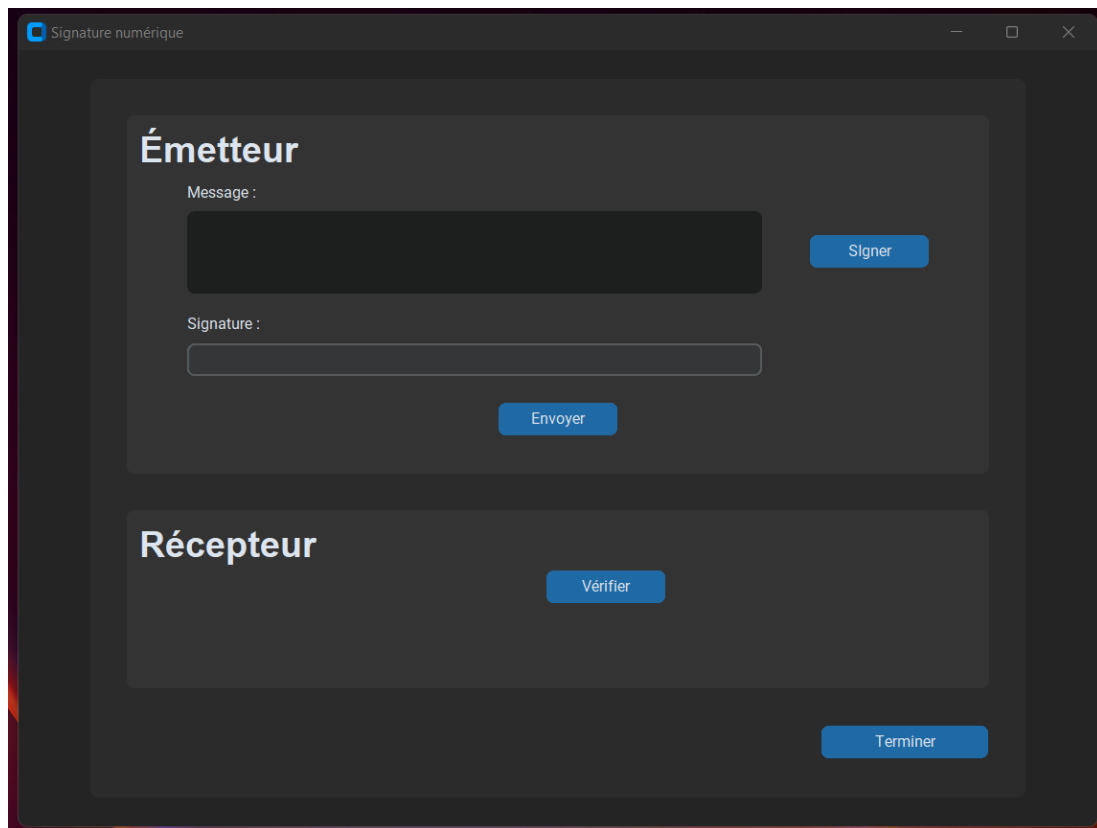
Mais, il y a plusieurs attaques plus sophistiquées contre le RSA, telles que les attaques par factorisation, les attaques par recherche de clé faible ou les attaques par oracle. Pour cette raison, il est important de suivre les bonnes pratiques de sécurité, telles que la gestion appropriée des clés, la mise en œuvre correcte de l'algorithme et la protection de la clé privée, pour garantir la sécurité de l'utilisation de RSA.

## 2 – Réalisation de l'interface :

Nous avons décidé de réaliser l'interface en utilisant le langage de programmation Python.

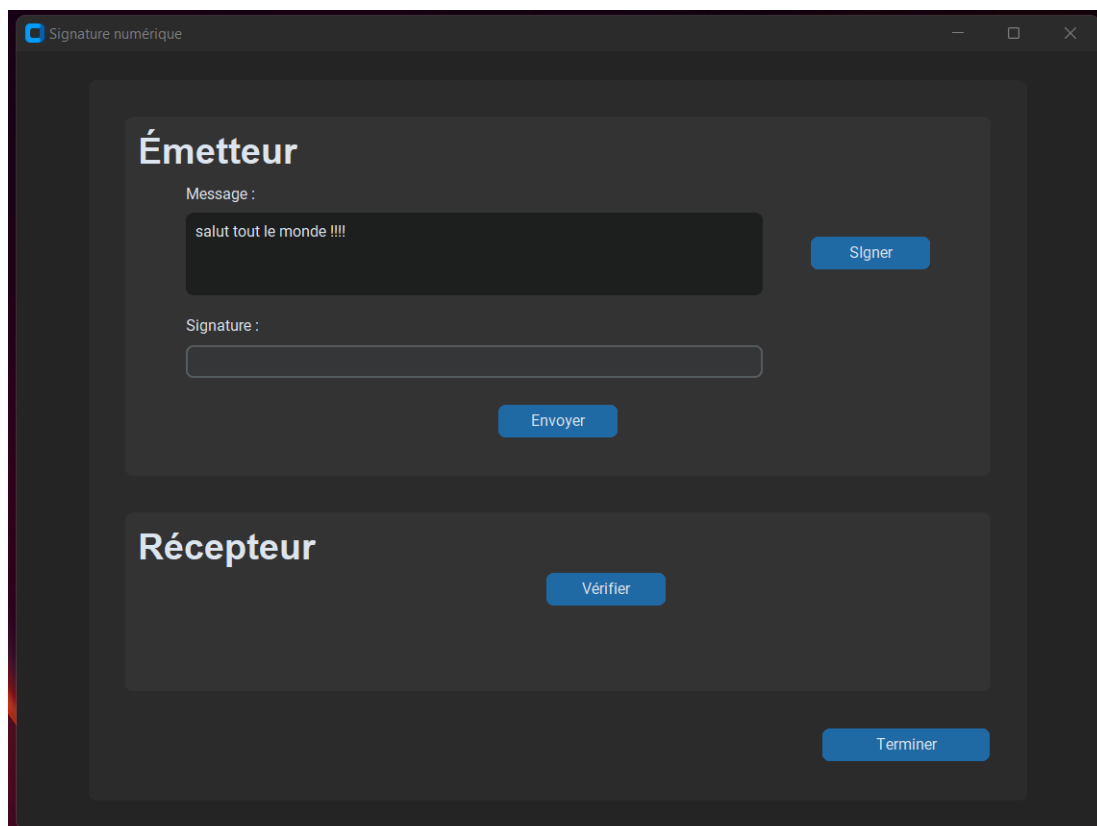
Cette interface nous permet de prendre une chaîne de caractères comme entrée puis il génère la signature associée à cette chaîne de caractère en utilisant la fonction de hachage SHA256 ainsi que l'algorithme RSA.

Il y a l'émetteur qui envoie une chaîne de caractères signée, et le récepteur qui la reçoit puis la vérifie.



The screenshot shows a window titled "Signature numérique". It contains two main sections: "Émetteur" and "Récepteur". In the "Émetteur" section, there is a "Message :" label above a text input field, a "Signer" button to its right, and a "Signature :" label above another text input field. Below these is an "Envoyer" button. The "Récepteur" section has a "Vérifier" button. At the bottom right of the window is a "Terminer" button.

L'émetteur commence par saisir la phrase « salut tout le monde !!!! » dans le champ de saisie.



This screenshot shows the same application window as before, but the text input field under "Message :" now contains the text "salut tout le monde !!!!". The "Signer" button remains to the right, and the "Signature :" field is still empty. The "Envoyer", "Vérifier", and "Terminer" buttons are also visible in their respective positions.

Puis en cliquant sur « signer » la signature se génère avec un message affiché à gauche affirmant « signature générée ! ».

**Signature numérique**

### Émetteur

Message :

salut tout le monde !!!!

Signer

Signature :

ee981d32bb540329991629642e8efe53ba9b556ab62425ccdc5a6e024589c2abfbf

Signature générée !

Envoyer

### Récepteur

Vérifier

Terminer

Ensuite, pour envoyer le message signé au récepteur on clique sur « Envoyer » et on remarque l'apparition du phrase « Bien reçu ! » dans le champs du récepteur.

**Signature numérique**

### Émetteur

Message :

salut tout le monde !!!!

Signer

Signature :

ee981d32bb540329991629642e8efe53ba9b556ab62425ccdc5a6e024589c2abfbf

Signature générée !

Envoyer

### Récepteur

Bien reçu !

Vérifier

Terminer

Par ailleurs, le récepteur peut vérifier la signature et le message qui les a reçu en cliquant sur « vérifier », si la signature correspond bien à l'émetteur la phrase « La signature est valide. Le message est authentique. » sera affichée.

The screenshot shows a web application titled "Signature numérique". It is divided into two main sections: "Émetteur" (Sender) and "Récepteur" (Receiver).

**Émetteur section:**

- Message :** A text input field containing "salut tout le monde !!!!".
- Signer:** A blue button to the right of the message field.
- Signature :** A text input field containing a long alphanumeric string: "ee981d32bb540329991629642e8efe53ba9b556ab62425ccdc5a6e024589c2abfbf".
- Signature générée !:** Text to the right of the signature field.
- Envoyer:** A blue button below the signature field.

**Récepteur section:**

- Bien reçu !:** Text displayed above the verification button.
- Vérifier:** A blue button.
- La signature est valide. Le message est authentique.:** Text displayed below the verification button.
- Terminer:** A blue button at the bottom right of the receiver section.

Dans le cas contraire, la phrase « La signature est invalide. Le message peut avoir été altéré. » sera affichée.

This screenshot shows the same "Signature numérique" application interface, but with a different result for the verification step.

**Émetteur section:**

- Message :** A text input field containing "salut tout le monde !!!".
- Signer:** A blue button to the right of the message field.
- Signature :** A text input field containing a long alphanumeric string: "7d1b3235ea33f2eea3118c0ea3cbbd699571ea1482da3a1d033d198dee8c8cbd19".
- Signature générée !:** Text to the right of the signature field.
- Envoyer:** A blue button below the signature field.

**Récepteur section:**

- Bien reçu !:** Text displayed above the verification button.
- Vérifier:** A blue button.
- La signature est invalide. Le message peut avoir été altéré.:** Text displayed below the verification button.
- Terminer:** A blue button at the bottom right of the receiver section.

Et finalement, le bouton « Terminer » sert à fermer l'interface.

## Références :

<https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>

<https://jafwin.com/2019/06/08/le-protocole-rsa-peut-etre-casse/>

[https://www.tutorialspoint.com/cryptography/cryptography\\_digital\\_signatures.htm](https://www.tutorialspoint.com/cryptography/cryptography_digital_signatures.htm)

## Annexe :

Notre interface codée en Python :

```
e.py > ...
1  import customtkinter
2  from tkinter import *
3  from Crypto.Hash import SHA256
4  from Crypto.PublicKey import RSA
5  from Crypto.Signature import pkcs1_15
6
7
8  # Générer une paire de clés publique/privée RSA pour l'émetteur
9  emetteur_key = RSA.generate(2048)
10
11 # Générer une paire de clés publique/privée RSA pour le récepteur
12 recep_teur_key = RSA.generate(2048)
13
14 customtkinter.set_appearance_mode("dark") # Modes: "System" (standard), "Dark", "Light"
15 customtkinter.set_default_color_theme("blue") # Themes: "blue" (standard), "green", "dark-blue"
16
17 app = customtkinter.CTk()
18 app.geometry("900x700")
19 app.title("Signature numérique")
20
21 frame = customtkinter.CTkFrame(master=app, width=900, height=600)
22 frame.pack(pady=20, padx=60, expand=True)
23
24 frame_21 = customtkinter.CTkFrame(master=frame, width= 720, height=300)
25 frame_21.place(x=30, y=30)
26
27 frame_22 = customtkinter.CTkFrame(master=frame, width= 720, height=150)
28 frame_22.place(x=30, y=360)
29
e.py > ...
30 #####functions declaration#####
31
32 def close_window():
33     app.quit()
34
35 signature = ""
36
37 def signer() :
38     # Calculer l'empreinte du message
39     hash_object = SHA256.new(message.encode())
40     # Signer le message avec la clé privée de l'émetteur
41     signature_ = pkcs1_15.new(emetteur_key).sign(hash_object)
42     signature_value.set(signature_.hex())
43     signature = signature_
44
45     signe = customtkinter.CTkLabel(master=frame_21, justify=customtkinter.LEFT, text="Signature générée !")
46     signe.place(x=570, y=190)
47
48 hash_object_recu = ""
49
50 def envoyer() :
51     message_2 = customtkinter.CTkLabel(master=frame_22, justify=customtkinter.LEFT, text="Bien reçu !")
52     message_2.place(x=260, y=50)
53
54 def verifier() :
55     # Calculer l'empreinte du message
56     hash_object = SHA256.new(message.encode())
57
58     # Signer le message avec la clé privée de l'émetteur
59     signature = pkcs1_15.new(emetteur_key).sign(hash_object)
```

```

e.py > ...
60
61     # Envoyer le message et la signature au récepteur
62     message_et_signature = {'message': message, 'signature': signature}
63
64     # Vérifier la signature à l'aide de la clé publique de l'émetteur
65     message_recu = message_et_signature['message']
66     #message_recu += "."
67     signature_recue = message_et_signature['signature']
68     hash_object_recu = SHA256.new(message_recu.encode())
69
70     hash_object_recu = SHA256.new(message_recu.encode())
71     try:
72         pkcs1_15.new(emetteur_key.publickey()).verify(hash_object_recu, signature_recue)
73
74         verification = customtkinter.CTkLabel(master=frame_22, justify=customtkinter.LEFT, text="La signature
est valide. Le message est authentique.")
75         verification.place(x=220, y=100)
76
77     except (ValueError, TypeError):
78
79         verification = customtkinter.CTkLabel(master=frame_22, justify=customtkinter.LEFT, text="La signature
est invalide. Le message peut avoir été altéré.")
80         verification.place(x=200, y=100)
81
82     ##### frame 21 #####
83
84     titre_e = customtkinter.CTkLabel(master=frame_21, justify=customtkinter.LEFT, text="Émetteur", font=
("Helvetica", 30, "bold"))
85     titre_e.place(x=10, y=10)
86
87     message_1 = customtkinter.CTkLabel(master=frame_21, justify=customtkinter.LEFT, text="Message :")

```

```

e.py > ...
87     message_1 = customtkinter.CTkLabel(master=frame_21, justify=customtkinter.LEFT, text="Message :")
88     message_1.place(x=50, y=50)
89
90
91     msg1 = customtkinter.StringVar()
92     msg_1 = customtkinter.CTkTextbox(master=frame_21, width=480, height=70)
93     msg_1.place(x=50, y=80)
94
95     message = msg_1.get("1.0", "end").strip()
96     # Envoyer le message et la signature au récepteur
97     message_et_signature = {'Message': message, 'signature': signature}
98
99     s_1 = customtkinter.CTkLabel(master=frame_21, justify=customtkinter.LEFT, text="Signature :")
100     s_1.place(x=50, y=160)
101
102
103     signature_value = customtkinter.StringVar()
104     signature_v = customtkinter.CTkEntry(master=frame_21, width=480, textvariable=signature_value)
105     signature_v.place(x=50, y=190)
106
107
108     button_generer1 = customtkinter.CTkButton(master=frame_21, text="Signer", width=100, command= signer)
109     button_generer1.place(x=570, y=100 )
110
111     button_generer1 = customtkinter.CTkButton(master=frame_21, text="Envoyer", width=100, command=envoyer)
112     button_generer1.place(x=310, y=240)

```

```

e.py > ...
115     ##### frame 22 #####
116
117     titre_r = customtkinter.CTkLabel(master=frame_22, justify=customtkinter.LEFT, text="Récepteur", font=
("Helvetica", 30, "bold"))
118     titre_r.place(x=10, y=10)
119
120     button_verifier = customtkinter.CTkButton(master=frame_22, text="Vérifier", width=100, command=verifier)
121     button_verifier.place(x=350, y=50 )
122
123     #####
124
125     button_close = customtkinter.CTkButton(master=frame, text="Terminer", command=close_window)
126     button_close.place(x=610, y=540)
127
128     app.mainloop()

```