PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

Criptografía y Seguridad Computacional - IIC3253
Tarea 2
Plazo de entrega: Viernes 9 de junio, 20hrs

Instrucciones

Cualquier duda sobre la tarea se deberá hacer en los *issues* del repositorio del curso. Si quiere usar alguna librería en sus soluciones debe preguntar primero si dicha librería está permitida. El foro es el canal de comunicación oficial para todas las tareas.

Configuración inicial. Para esta tarea utilizaremos *github classroom*. Para acceder a su repositorio privado debe ingresar al siguiente link, seleccionar su nombre y aceptar la invitación. El repositorio se creará automaticamente una vez que hagas esto y lo podrás encontrar junto a los repositorios del curso. Para la corrección se utilizará Python 3.10.

Entrega. Al entregar esta tarea, su repositorio se deberá ver exactamente de la siguiente forma:

```
Repositorio
├── Pregunta_1
│   ├── requirements.txt
│   ├── pregunta1_a.ipynb
│   └── 1_b_c.pdf
├── Pregunta_2
│   ├── requirements.txt
│   └── pregunta2.ipynb
├── .gitignore
├── README.md
└── .git
```

Deberá considerar lo siguiente:

- El archivo `requirements.txt` dentro de la carpeta de una pregunta deberá especificar todas las librerías que se necesitan instalar para ejecutar el código de su respuesta a dicha pregunta. Este archivo debe seguir la especificación de Pip, es decir se debe poder ejecutar el comando `pip install -r requirements.txt` suponiendo una versión de `Pip` mayor o igual a 22.0 que apunte a la versión 3.10 de Python. Si su respuesta no requiere librerías adicionales, este archivo debe estar vacío (pero debe estar en su repositorio).

- Se recomienda utilizar tipado para las variables y funciones en Python.
- La solución de cada problema de programación debe ser entregada como un Jupyter Notebook (esto es, un archivo con extensión `ipynb`). Este archivo debe contener comentarios que expliquen claramente el razonamiento tras la solución del problema, idealmente utilizando *markdown*. Más aun, su archivo deberá ser exportable a un módulo de Python utilizando el comando de consola

  ```
  jupyter nbconvert --to python preguntaX.ipynb
  ```

  Este comando generará un archivo `preguntaX.py`, del cual se deben poder importar las funciones y clases que se piden en cada pregunta. Por ejemplo, luego de ejecutar este comando, se debe poder importar desde otro archivo Python (ubicado en el mismo directorio) la clase `Receiver` simplemente con `from pregunta1_a import Receiver`.
- Para cada problema cuya solución se deba entregar como un documento (en este caso la preguntas 1), usted deberá entregar un archivo `.pdf` que, o bien fue construido utilizando LaTeX, o bien es el resultado de digitalizar un documento escrito a mano. En caso de optar por esta última opción, queda bajo su responsabilidad la legibilidad del documento. Respuestas que no puedan interpretar de forma razonable los ayudantes y profesores, ya sea por la caligrafía o la calidad de la digitalización, serán evaluadas con la nota mínima.

Preguntas

1. En esta pregunta used va a implementar y demostrar la corrección de un esquema criptográfico que utiliza claves más simples que las de RSA, y que además es aleatorizado, produciendo con una alta probabilidad cifrados distintos si un mensaje es encriptado más de una vez.

   Para definir este esquema, necesitamos introducir un poco de notación. Dado un número natural n, sea $\#\text{Bit}(n)$ el número de bits en la representation binaria de n. Además, dados dos números naturales n, m tales que $m > 0$, sea

   $$\text{Div}(n, m) \quad = \quad \left\lfloor \frac{n-1}{m} \right\rfloor.$$

   Entonces, la clave pública P_A y la clave secreta S_A de un usuario A son generadas de la siguiente forma.

   (a) Genere dos números primos distintos P y Q tales que $P \geq 3$, $Q \geq 3$ y $\#\text{Bit}(P) = \#\text{Bit}(Q)$. Sea $N = P \cdot Q$ y $\phi(N) = (P-1) \cdot (Q-1)$.

   (b) Defina $P_A = N$ y $S_A = \phi(N)$.

   La función de cifrado Enc_{P_A} es definida de la siguiente forma. Dado un mensaje $m \in \{0, \ldots, N-1\}$, se genera al azar un número $r \in \{1, \ldots, N-1\}$ tal que $MCD(r, N) = 1$, y se construye

   $$Enc_{P_A}(m) \quad = \quad ((N+1)^m \cdot r^N) \mod N^2$$

   La función de descifrado Dec_{S_A} es definida de la siguiente forma. Sea $B \in \{0, \ldots, N-1\}$ el inverso de $\phi(N)$ en módulo N, vale decir, B satisface la condición

   $$\phi(N) \cdot B \equiv 1 \mod N$$

   Entonces dado un texto cifrado $c \in \{0, \ldots, N^2-1\}$, se define

   $$Dec_{S_A}(c) \quad = \quad \left[\text{Div}(c^{\phi(N)} \mod N^2,\, N) \cdot B\right] \mod N$$

   Responda las siguientes preguntas, en las cuales va a implementar el esquema criptográfico y va a demostrar que es correcto.

   (a) Implemente el esquema criptográfico definido en esta pregunta completando el Jupyter notebook `pregunta1_a.ipynb`. Para que su pregunta sea considerada correcta, su notebook deberá correr de principio a fin habiendo completado los metodos marcadas con `##### POR COMPLETAR`. Las entradas y salidas de estos métodos no pueden ser modificadas, pero sí puede agregar métodos adicionales si los considera necesarios. Se evaluará con un programa externo la implementación de sus clases `Receiver` y `Sender`.

   (b) Demuestre que $MCD(N, \phi(N)) = 1$. Nótese que de esto se deduce la existencia del número B, que es el inverso de $\phi(N)$ en módulo N.

   (c) Dado $m \in \{0, \ldots, N-1\}$, demuestre que:

   $$Dec_{S_A}(Enc_{P_A}(m)) \quad = \quad m$$

2. En esta pregunta deberá escribir un programa que verifique la autenticidad de un Json Web Token (JWT) en sus variantes HS256 y RS256. Para esto, deberá escribir un Jupyter notebook siguiendo las instrucciones explicadas arriba que al menos defina la siguiente función:

```python
def validate_jwt(jwt: str, key: str) -> bool:
    """
    Arguments:
      jwt: a well-formed Json Web Token
      key: the key to verify the validity of the jwt
    Returns:
      valid: is the jwt is valid w.r.t the provided key?
    """
```

Su notebook podrá definir funciones auxiliares que ayuden a simplificar la lectura. Deberá estar explicado y ser fácil de seguir para una persona que entiende los contenidos del curso.

Importante: Su notebook sólo podrá importar las siguientes librerías externas

```python
from hashlib import sha256
from base64 import urlsafe_b64decode, urlsafe_b64encode
```

Para validar un JWT que usa HS256, el parámetro `key` es un string que representa la llave a usar en HMAC-SHA256. Por ejemplo el siguiente JWT es válido con la llave `IIC3253`.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjbGFzcyI6IkNyaXB0b2dy
YWbDrWEgeSBTZWd1cmlkYWQgQ29tcHV0YWNpb25hbCIsInVuaXZlcnNpdHki0
iJQVUMgQ2hpbGUifQ.Cn1AACqINaUTbAJuh_V4lBcr9X4dRp8FUX9sGDkX-Ss
```

Para un JWT que usa RS256, el parámetro `key` es una llave pública RSA en formato `PKCS#1` para validar la firma RSA contenida en el JWT. Por ejemplo el JWT

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiQ3JpcHRvZ3JhZsOt
YSB5IFNlZ3VyaWRhZCBDb21wdXRhY2lvbmFsIiwiY291cnNlIjoiSUlDMzI1MyIsI
nVuaXZlcnNpdHkiOiJQVUMgQ2hpbGUifQ.CmoMizX-_E2Ugd7-JDThCfrXTJbg38W
Lal3HipmnA8oAUh1yG9IU1n_klJkmPIT3knxrmrJMXxh6gTCOylLQfKSQI7pHsYUr
-yOd5gL7XpnT3stvOtYDO383cBnrL5X8EV01lUxJJenYG5Qr4uVG7Msg-4fUJbTqT
R2t0Jx2UQ2pfi_jxgfg6lAjSLK9TygntJJ-eJV0Q8IipVYnqtCxBS-OIXekalyjpB
Hksf_ibiJtPrMJI3Kvyj3dwrETth8c4yg2IIh22uoJHrJArNk3xPfeSsasZT0ixfM
E8Mlnkd4HwpbcNZZl-FpsBPbPWHKynZXptq8uS65PxKmTggg8kA
```

es válido con la llave pública

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAu1SU1LfVLPHCozMxH2Mo
4lgOEePzNm0tRgeLezV6ffAt0gunVTLw7onLRnrq0/IzW7yWR7QkrmBL7jTKEn5u
+qKhbwKfBstIs+bMY2Zkp18gnTxKLxoS2tFczGkPLPgizskuemMghRniWaoLcyeh
kd3qqGElvW/VDL5AaWTg0nLVkjRo9z+40RQzuVaE8AkAFmxZzow3x+VJYKdjykkJ
0iT9wCS0DRTXu269V264Vf/3jvredZiKRkgwlL9xNAwxXFg0x/XFw005UWVRIkdg
cKWTjpBP2dPwVZ4WWC+9aGVd+Gyn1o0CLelf4rEjGoXbAAEgAqeGUxrcIlbjXfbc
mwIDAQAB
-----END PUBLIC KEY-----
```