Proyecto 1

1. Introducción

Este proyecto está relacionado con la implementación de un esquema de firma digital post-cuántico basado en el **LUOV (Lifted Unbalanced Oil and Vinegar)**, un criptosistema multivariado. El objetivo es implementar la generación de llaves (Key Generation) del esquema LUOV, con soporte para distintos parámetros recomendados. El proyecto también incluye pruebas unitarias para verificar el correcto funcionamiento de los componentes.

2. Requerimientos

El archivo requirements.txt define las dependencias necesarias para el proyecto:

numpy==1.23.0

pytest==7.1.2

hashlib

Estas son bibliotecas estándar, donde:

- **numpy** se utiliza para manejo eficiente de matrices.
- **pytest** para pruebas unitarias.
- **hashlib** para funciones de hashing criptográfico(requirements).

3. Archivos Principales

keygen.py

Este archivo contiene el código principal para la **generación de llaves** (Key Generation) del esquema LUOV. De acuerdo con la especificación del algoritmo LUOV(LUOV Specification), la función de generación de llaves crea una **clave pública** y una **clave privada** utilizando una semilla pseudoaleatoria. Esta semilla es utilizada para generar los coeficientes del polinomio público (P) y las transformaciones lineales necesarias.

Este archivo contiene el código principal para la **generación de llaves** (Key Generation) del

esquema LUOV. El siguiente fragmento de código ilustra cómo se implementa el proceso de

generación de claves:

```
def keygen():
 private_seed = get_random_seed()
 public_seed, T = generate_public_seed_and_T(private_seed)
 C, L, Q1 = generate_public_map(public_seed)
 Q2 = findQ2(Q1, T)
 return (public_seed, Q2), private_seed
```

constants.py

Define valores constantes utilizados en el esquema. Estas constantes pueden incluir los parámetros de campo finito y tamaños específicos de las claves para los diferentes conjuntos de parámetros de LUOV. Los parámetros son cruciales para la seguridad y el tamaño de las firmas digitales, por lo que definirlos correctamente garantiza la robustez del sistema(LUOV Specification).

utils.py

Este archivo incluye varias **funciones auxiliares** que permiten realizar operaciones como generación de matrices o cálculos de hash. Estas funciones se utilizan dentro del proceso de generación de llaves para manejar las operaciones de álgebra lineal y cálculo de hash, necesarios para obtener los coeficientes de la clave pública a partir de la semilla(LUOV Specification).

```
def generate_public_map(seed):

 # Generate coefficients for the public key polynomial

 C = hash_function(seed)

 L = linear_map(seed)

 Q1 = quadratic_map(seed)

 return C, L, Q1
```

utils_for_verify.py

Aquí se encuentran las funciones que se utilizan para verificar las firmas generadas. Estas funciones probablemente incluyen la evaluación de las ecuaciones del polinomio público sobre una firma y un mensaje dado, asegurándose de que el resultado coincida con el hash del mensaje, lo que garantiza la integridad y autenticidad del mismo(LUOV Specification).

__init_.py

Este archivo podría estar inicializando los módulos del paquete, definiendo qué funciones o clases son accesibles desde el exterior del módulo. Generalmente se utiliza para importar las funciones claves del proyecto en un paquete(LUOV Specification).

test_keygen.py

Este archivo contiene **pruebas unitarias** utilizando pytest, verificando la correcta implementación de la generación de llaves. Las pruebas están diseñadas para asegurar que los resultados de la generación de llaves sean válidos según las especificaciones de LUOV y que se comporten de manera consistente(requirements)(LUOV Specification).

```
def test_keygen():

 (public_key, private_key) = keygen()
 assert public_key is not None

 assert private_key is not None
```

4. Explicación de los Componentes del Algoritmo LUOV

Algoritmo de Generación de Claves

La generación de llaves en LUOV sigue estos pasos generales:

1. **Semilla privada**: Se utiliza una semilla privada para generar una transformación lineal y parte de los coeficientes del polinomio público.
2. **Transformación T**: Esta es una matriz que transforma los valores de las variables de vinagre y aceite.
3. **Coeficientes del polinomio público**: Se generan usando funciones hash extendidas como SHAKE128 o SHAKE256.
4. **Clave pública y privada**: La clave pública incluye una semilla pública y una parte del polinomio público, mientras que la clave privada simplemente almacena la semilla privada.

Este proceso asegura que la clave privada sea compacta y la clave pública, aunque grande, sea computacionalmente eficiente para firmar y verificar(Proyecto 1)(LUOV Specification).

Firma Digital

El proceso de firma en LUOV implica:

1. Generar un hash del mensaje a firmar.
2. Resolver el sistema de ecuaciones cuadráticas asociado con el polinomio público para encontrar una firma que satisfaga la ecuación.
3. Usar una combinación de las variables de vinagre y aceite para crear una firma válida que será verificada por el destinatario.

Verificación de Firmas

La verificación consiste en calcular el polinomio público con la firma proporcionada y verificar si su salida coincide con el hash del mensaje original. Si coinciden, la firma es válida (LUOV Specification).

5. Conclusión

El esquema de firma LUOV, siendo post-cuántico, ofrece una solución robusta contra ataques utilizando computación cuántica. La implementación en Python permite una fácil adaptación y extensión, asegurando que se puedan manejar diferentes conjuntos de parámetros de manera flexible.