Proyecto 3 Criptografía

Integrantes:

Brancys Barrios

Nathalia De La Rans

Diego Linero

Docente:

Eduardo Angulo Madrid

Barranquilla

Universidad del Norte

26/11/2024

• Introducción

Finalizando la continuación de esta serie de laboratorios, en el Algoritmo 8 consolidamos el aprendizaje sobre las firmas digitales que hemos explorado previamente en los algoritmos 4 y 6. Gracias a lo trabajado como grupo, hemos afinado nuestro entendimiento sobre la importancia de las firmas digitales en garantizar la autenticidad e integridad de los datos en entornos digitales. Este último ejercicio nos permitió integrar estrategias criptográficas avanzadas y fortalecer nuestro conocimiento en esquemas seguros y resistentes a ataques.

A lo largo de esta experiencia, comprendimos cómo los conceptos fundamentales de las firmas digitales se aplican en diferentes escenarios, con énfasis en la generación, verificación y resistencia ante posibles vulnerabilidades, considerando también la viabilidad en un mundo post-cuántico.

• Funcionamiento esquemático firma digital.

En el **Algoritmo 8**, desarrollamos un esquema de firma digital basado en LUOV, un sistema criptográfico multivariante diseñado para resistir ataques cuánticos. Este esquema asegura su fortaleza criptográfica mediante la dificultad inherente de resolver sistemas de ecuaciones polinomiales multivariadas sobre campos finitos, haciéndolo altamente relevante en un entorno post-cuántico.

La generación de firmas en este esquema aprovecha propiedades matemáticas que permiten garantizar autenticidad e integridad, combinando estrategias avanzadas con un diseño eficiente para mantener un equilibrio entre seguridad y desempeño.

Generación de llaves

El proceso de generación de claves comienza con la creación de una **semilla privada** que alimenta un generador de números pseudoaleatorios. Este generador produce tanto la **semilla pública** como la **matriz** T, un componente esencial para ocultar la estructura del mapa secreto F.

A partir de la semilla pública, se derivan tres elementos fundamentales de la llave pública:

1. **Parte Constante (C)**: Representa un desplazamiento inicial del mapa polinomial.
2. **Parte Lineal (L)**: Introduce la linealidad necesaria para la construcción del sistema.
3. **Parte Cuadrática ($Q1$)**: Incluye los términos no lineales que refuerzan la complejidad.

Posteriormente, se calcula una segunda parte cuadrática ($Q2$), que junto con $Q1$ y la matriz TTT, conforma el núcleo del mapa público. Finalmente, la **clave pública** se publica como $Q2$ y la semilla pública, mientras que la **clave privada** permanece como la semilla inicial utilizada para generar todos los componentes.

• Códigos desarrollados y criptosistemas utilizados

Nosotros continuamos el desarrollo del esquema de firma digital avanzando al algoritmo 8, basándonos en el trabajo previo con los algoritmos 4 y 6. En nuestra evolución del **Algoritmo 4** al **Algoritmo 8**, el proceso de implementación del esquema LUOV experimentó varias mejoras y optimizaciones. Desde el principio, nos centramos en la eficiencia de las operaciones matemáticas y criptográficas, adaptando el código para manejar grandes volúmenes de datos de manera fluida. Esta transición implicó ajustar varios aspectos del diseño y estructura del código, permitiéndonos abordar más eficazmente los desafíos de seguridad en un entorno post-cuántico.

Estrategias y diseño del código

El **Algoritmo 8** tiene como núcleo el **algoritmo de generación de llaves (KeyGen)**, que ha sido implementado para seguir de cerca la especificación oficial del esquema LUOV. A lo largo de esta transición desde el **Algoritmo 4**, mejoramos la modularidad del código, dividiendo el proceso en tareas claras, cada una encargada de una parte específica, como la generación de las llaves y la firma de mensajes. La optimización de este proceso fue crucial, ya que teníamos que garantizar que pudiera manejar la complejidad matemática asociada con los polinomios multivariantes de manera eficiente.

La **generación de claves** sigue un proceso similar al del **Algoritmo 4**, pero con importantes optimizaciones, especialmente en la creación y manejo de la matriz T. En lugar de depender de algoritmos tradicionales de clave pública, utilizamos un generador de números pseudoaleatorios junto con la función hash **SHAKE256**, la cual

se había optimizado para generar salidas de longitud variable. Esto fue crucial para generar no solo la matriz T, sino también para derivar los polinomios $Q1$ y $Q2$ del mapa cuadrático, que forman el núcleo de la clave pública.

```python
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.scrypt import Scrypt

# Función hash SHAKE256 para generar la semilla y la matriz T
Tabnine | Edit | Test | Explain | Document | Ask
def shake256_generator(seed, length=64):
    shake = hashes.Hash(hashes.SHAKE256())
    shake.update(seed)
    return shake.finalize()[:length]  # Generación de bits de longitud variable
```

En el **Algoritmo 8**, también optimizamos el manejo de los polinomios $Q1$ y $Q2$. Estos componentes son fundamentales para el funcionamiento del esquema de firma, y aseguramos que las funciones matemáticas y su codificación fueran lo suficientemente robustas como para resistir ataques cuánticos. Utilizamos la función **SHAKE256** para generar estos polinomios, lo que permitió asegurar la aleatoriedad y seguridad en la creación de estos componentes clave.

Criptosistemas utilizados

En nuestra evolución desde el **Algoritmo 4** hacia el **Algoritmo 8**, seleccionamos las funciones hash y criptosistemas adecuados para asegurar que nuestro sistema fuera seguro y eficiente. En el **Algoritmo 8**, optamos por **SHAKE256** como el mecanismo principal para generar semillas y derivar componentes clave, ya que ofrece una excelente resistencia a ataques de colisión y tiene la capacidad de generar salidas de longitud variable, lo que la hace ideal para un sistema criptográfico post-cuántico.

El algoritmo **SHAKE256** fue crucial para la transición del **Algoritmo 4** al **Algoritmo 8**, permitiéndonos no solo generar la matriz TTT, sino también derivar los polinomios $Q1$ y $Q2$ con gran eficiencia. Esto fue un cambio significativo en nuestra implementación, ya que utilizamos un esquema criptográfico que, a su vez, permitió mejorar la eficiencia en la generación de las llaves públicas y privadas.

```python
# Implementación de SHAKE256 para generar componentes de la clave
Tabnine | Edit | Test | Fix | Explain | Document | Ask
def generate_public_key(seed):
    # Derivación de la semilla pública y la matriz T usando SHAKE256
    seed_bytes = shake256_generator(seed)
    T_matrix = seed_bytes[:64]  # Ejemplo de extracción de la matriz T
    Q1 = seed_bytes[64:128]  # Polinomio cuadrático Q1
    Q2 = seed_bytes[128:192]  # Polinomio cuadrático Q2
    return Q1, Q2, T_matrix

# Generar la clave pública
```

Este avance no solo mejoró la seguridad del sistema, sino que también optimizó el rendimiento, ya que ahora podíamos generar estas claves de forma más rápida y eficiente, algo que no estaba tan optimizado en el **Algoritmo 4**.

Desarrollo de la firma digital

El desarrollo de la firma digital en el algoritmo 8 fue una evolución natural de lo que habíamos logrado en el algoritmo 4 y 6. Como grupo, seguimos una estrategia iterativa y de optimización, donde nos centramos en la eficiencia y la seguridad en cada paso del proceso. El algoritmo que implementamos comenzó con la generación de una firma parcial, donde probábamos diferentes asignaciones para las variables de vinagre v, tal como lo hicimos en el desarrollo anterior, pero esta vez con una implementación más eficiente, aprovechando los avances en la construcción de las matrices y en la optimización del tiempo de ejecución.

El paso clave fue, nuevamente, cuando las variables de vinagre se fijaban, lo que reducía el sistema cuadrático a un sistema lineal. Esto permitió que la resolución de ecuaciones se manejara con mayor rapidez, utilizando álgebra matricial de forma más eficiente. Aquí es donde el código del algoritmo 8 realmente demostró sus mejoras, al haber optimizado tanto la forma en que manejábamos las matrices aumentadas como el uso de la eliminación gaussiana para obtener las soluciones de las variables de aceite o. Este refinamiento se logró gracias al aprendizaje y los cambios que hicimos a lo largo de los diferentes algoritmos, especialmente durante las fases de prueba y ajuste en los pasos previos.

Como grupo, también nos dimos cuenta de que aplicar la matriz T inversa para recuperar el mensaje original era un paso crucial, y necesitábamos hacerlo con la mayor eficiencia posible para no comprometer el rendimiento. Esto nos llevó a usar

nuevamente técnicas de precomputación para reducir el tiempo de cálculo en cada paso, especialmente en la parte de verificación de firmas, lo cual era un desafío dado el tamaño de los datos y la complejidad del sistema en este algoritmo.

Bibliotecas utilizadas

Como se ha mencionado anteriormente, utilizamos SHAKE256 de la biblioteca pycryptodome, que es clave en todo el esquema LUOV por su capacidad para generar hashes seguros de longitud variable. Además, empleamos otras librerías como binascii, base64, json, os y numpy para la manipulación de datos binarios, generación de valores aleatorios y operaciones matemáticas necesarias para resolver los sistemas de ecuaciones.

Desafíos y aprendizajes

Los desafíos que enfrentamos al desarrollar la firma digital en el algoritmo 8 fueron, en muchos sentidos, una continuación de los que ya habíamos identificado en el algoritmo 4 y 6. El problema principal seguía siendo la complejidad computacional, sobre todo cuando se trataba de resolver los sistemas de ecuaciones involucrados. A medida que avanzábamos, nos dimos cuenta de que necesitábamos un enfoque más avanzado para optimizar el tiempo de ejecución sin comprometer la seguridad del sistema.

Así, implementamos nuevas estrategias de precomputación que nos permitieron reducir el tiempo de cálculo en la verificación de firmas. En lugar de recalcular ciertos componentes cada vez que verificábamos una firma, almacenamos y reutilizamos estos cálculos, lo que resultó en una mejora significativa en la eficiencia. Este enfoque fue un claro ejemplo de cómo, como grupo, fuimos capaces de aplicar los aprendizajes previos del laboratorio a lo largo de los algoritmos para construir un sistema más robusto y eficiente.

En resumen, al avanzar del algoritmo 4 al algoritmo 8, no solo mejoramos la eficiencia y seguridad del sistema de firma digital, sino que también aprendimos a modularizar y optimizar el código de manera más efectiva. Esto se logró gracias al esfuerzo conjunto del grupo y la iteración constante de cada código desarrollado, lo que nos permitió crear una implementación del sistema LUOV más escalable y adaptable a los desafíos de seguridad post-cuántica.