Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

IMPLEMENTACIÓN DE CIFRADO HOMOMÓRFICO EN UNA RED DE MEZCLA

Axel Ernesto Moreno Cervantes
Instituto Politécnico Nacional
amorenoc@ipn.mx

Verónica Agustín Domínguez
Instituto Politécnico Nacional
vagustin@ipn.mx

Benjamín Cruz Torres
Instituto Politécnico Nacional
bcruzt@ipn.mx

Abstract

En este trabajo se presenta el análisis de los algoritmos de cifrado homomórfico de ElGamal y de Paillier, con el objetivo de utilizar alguno de los algoritmos en una red de mezcla orientada a los sistemas de voto electrónico. De esta manera, se aprovecha la principal propiedad del cifrado homomórfico, la cual permite manipular los datos cifrados sin la necesidad de que estos sean descifrados. De esta manera se busca proveer anonimato a la información, al mantener la identidad del usuario desconocida.

Palabras clave: Criptografía, cifrado homomórfico, anonimato

La criptografía forma parte de nuestra vida diaria en muchas aplicaciones que se utilizan, aunque los algoritmos de cifrado estándar no siempre son la mejor opción. Por ejemplo, en los sistemas de voto electrónico se necesita brindar anonimato a los votantes y solo conocer el total de votos recibidos por los usuarios, esto garantiza que la votación sea secreta (Ramchen, 2012). El cifrado homomórfico permite realizar operaciones sobre datos cifrados sin la necesidad de tener que descifrar los datos previamente y con ello se puede mantener la confidencialidad de la información durante el tratamiento de ésta, además de disminuir las operaciones de cifrado y descifrado que agregarían un costo computacional adicional y significan puntos de falla en los que pueda existir una fuga de información (Pigatto, 2012).

Respecto al anonimato, se distingue entre seudonimato y anonimato, en el primero se utiliza un seudónimo que se asocia con un identificador único manteniendo la identidad real detrás, mientras que el anonimato no hay información identificable de un objeto o persona (Oujani, 2011). El uso de seudónimos es una estrategia común al momento de proveer anonimato.

Sin embargo, esta estrategia no puede evitar que se rastreen los pasos del usuario y sea posible determinar la identidad detrás. Una alternativa para proveer anonimato es el uso de redes de mezcla, las cuales utilizan una llave

pública y un reenviador de confianza, la llave pública es un seudónimo persistente y es el bloque básico para la construcción en sistemas de anonimato modernos. Nuestro trabajo se centra en el uso de redes de mezcla de re-cifrado, la cual consiste en servidores de mezcla que implementan el re-cifrado de la lista de mensajes recibidos junto a la permutación de estos para que la salida sea en orden aleatorio. Después de pasar por todos los servidores, la lista de textos cifrados es descifrada y el resultado es una lista de los mensajes en orden aleatorio. De esta manera se brinda irrastreabilidad, debido a que es imposible determinar la posición de cada elemento en la lista inicial, dando a los participantes anonimato (Larsson & Svensson). En la Figura 1 se muestra un ejemplo de red de mezcla de re-cifrado en la cual los cuadros a la izquierda representan los mensajes cifrados con la llave pública de los reenviadores que son los servidores de enrutamiento.

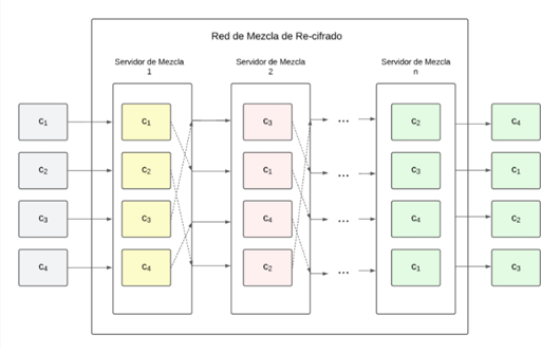

Figura 1. Red de Mezcla de re-cifrado [elaboración propia].

La relación de la propiedad homomórfica y las redes de mezcla de re-cifrado ocurre al momento de aplicar el re-cifrado en el servidor de mezcla. Por ejemplo, sea $Ekpub$ la función de cifrado de Paillier y sea m el mensaje original. Entonces el servidor de mezcla recibe el mensaje cifrado y realiza el siguiente proceso para re-cifrar el mensaje:

$$Ekpub(m) \rightarrow Ekpub(m) * Ekpub(0)$$

La multiplicación en el grupo de los textos cifrados en Paillier corresponde a la adición en el grupo de los textos planos, como lo indica su propiedad homomórfica. Al multiplicar por 0 en texto cifrado se está realizando lo equivalente a sumar el elemento neutro al mensaje original $m + 0 = m$, por lo tanto, al momento de descifrar el resultado de la operación se obtiene el mensaje original.

I. Metodología

Para el desarrollo de este trabajo se siguieron los pasos del método científico con el fin de analizar los principales algoritmos de cifrado homomórfico (ElGamal y Paillier) y así determinar cuál de ellos utilizar en la implementación de la red de mezcla. Las implementaciones de ambos algoritmos se realizaron en Python.

En la implementación del algoritmo de ElGamal se utilizaron distintos módulos: un módulo para la generación de llaves, otro para el cifrado, el tercero en el descifrado, y el cuarto en la multiplicación de textos cifrados con el propósito de comprobar el homomorfismo.

En el módulo de generación de llaves se generan las llaves pública y privada. Para la llave privada se calcula el número primo p con un tamaño de 3072 bits y un número aleatorio $a \in Z_p^*$. Para la llave pública se obtiene el generador g, que por motivos de rendimiento se toma un número aleatorio, y β que se calcula de acuerdo con lo indicado en la Tabla 1.

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

Tabla 1. Algoritmo para generación de llaves de ElGamal [elaboración propia].

Algoritmo 1. Generación de llaves para el cifrado ElGamal
Salida: Llave pública, Llave privada
Seleccionar un número primo p, $\lvert p\rvert$ tal que $\lvert p\rvert \geq 3072$
Seleccionar un número $\propto$ tal que sea el generador de un grupo Z_p^*
Seleccionar un número aleatorio a tal que $1 \leq a \leq q-1$
$\beta \leftarrow \propto^a \ mod\ p$
Llave pública $\leftarrow (p, \propto, \beta)$
Llave privada $\leftarrow a$

Para el cifrado con ElGamal se utilizó el algoritmo que se muestra en la Tabla 2, este recibe como parámetros de entrada los valores de la llave pública p, g, β, y el mensaje en texto plano a cifrar x. La función devuelve los valores y_1 y y_2 que son los elementos de un texto cifrado en ElGamal.

Tabla 2. Algoritmo de cifrado de ElGamal [elaboración propia].

Algoritmo 2. Cifrado con ElGamal
Entrada: Llave pública, x
Salida: y_1, y_2
Seleccionar un número aleatorio k del grupo Z_{p-1}^*
$y_1 \leftarrow \propto^k \ mod\ p$
$y_2 \leftarrow x \times \beta^k \ mod\ p$

En el descifrado utilizando ElGamal se utilizó el algoritmo mostrado en la Tabla 3, éste recibe como parámetros de entrada los elementos del texto cifrado y_1 y y_2, y la llave privada a, p. En la salida obtenemos el texto plano original.

Tabla 3. Algoritmo de descifrado de ElGamal [elaboración propia].

Algoritmo 3. Descifrado con ElGamal
Entrada: Llave Privada, y_1, y_2
Salida: x
Seleccionar un número aleatorio k del grupo Z_{p-1}^*
$x \leftarrow y_2 \times (y_1^a)^{-1} \ mod\ p$

Para comprobar la propiedad homomórfica se utilizó el algoritmo mostrado en la Tabla 4, en este se aplica la operación de multiplicación sobre dos textos cifrados, recibe los elementos de un primer texto cifrado y_1 y y_2, así como los elementos de un segundo texto cifrado y_3 y y_4.

Tabla 4. Algoritmo de propiedad homomórfica de ElGamal [elaboración propia].

Algoritmo 4. Homomorfismo con cifrado ElGamal
Entrada: y_1, y_2, y_3, y_4
Salida: y'_1, y'_2
Seleccionar un número aleatorio k del grupo Z_{p-1}^*
$y'_1 \leftarrow y_1 \times y_3 \ mod\ p$
$y'_2 \leftarrow y_2 \times y_4 \ mod\ p$

En la implementación de Paillier se utilizaron los mismos módulos: generación de llaves, cifrado, descifrado y multiplicación de textos cifrados.

Para la generación de llaves de Paillier se usó el algoritmo mostrado en la Tabla 5, el cual devuelve la llave pública y la llave privada. Un tamaño de llave segura debe ser mayor a 2048 bits. En esta implementación utilizamos una llave de 3072 bits.

Tabla 5. Algoritmo para generación de llaves de Paillier [elaboración propia].

Algoritmo 5. Generación de llaves para el cifrado de Paillier
Salida: Llave pública, Llave privada
Seleccionar un número primo p, $\lvert p\rvert$ tal que $\lvert p\rvert \geq 3072$
Seleccionar un número q tal que q debe ser independiente de p
$n \leftarrow p \times q$
$\emptyset(n) \leftarrow (p-1) \times (q-1)$
Llave pública $\leftarrow (n)$
Llave privada $\leftarrow (n, \emptyset(n))$

Para el cifrado con Paillier se utilizó el algoritmo mostrado en la Tabla 6, este recibe como parámetro la llave pública n, así como el mensaje a cifrar x. Devuelve como salida el texto cifrado c.

Tabla 6. Algoritmo de cifrado de Paillier [elaboración propia].

Algoritmo 6. Cifrado con Paillier
Entrada: Llave pública, x
Salida: c
$c' \leftarrow (1+n)^x$
$r \leftarrow r^n$
$c \leftarrow (c' \times r') \ mod\ n^2$

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

En el descifrado con Paillier se utilizó el algoritmo que se muestra en la Tabla 7, para ello se recibe como entrada la llave privada n, $\phi(n)$, así como el mensaje cifrado c. Devuelve como salida el mensaje original x.

Tabla 7. Algoritmo de descifrado de Paillier [elaboración propia].

Algoritmo 7. Descifrado con Paillier
Entrada: Llave Privada, c
Salida: x
$c' \leftarrow (c^{\emptyset(n)} \bmod n^2) - 1$
$L \leftarrow c'//n$
$x \leftarrow (c' \times \emptyset(n)^{-1}) \bmod n$

Para la implementación del homomorfismo con Paillier se utilizó el algoritmo mostrado en la Tabla 8, donde se multiplican dos textos cifrados c_1 y c_2, así se obtiene un texto cifrado c equivalente a la suma de los mensajes en texto plano correspondientes a x_1 y x_2.

Tabla 8. Algoritmo de propiedad homomórfica de Paillier [elaboración propia].

Algoritmo 8. Homomorfismo con cifrado de Paillier
Entrada: $c1, c2$
Salida: c
$c \leftarrow c1 \times c2$

Para el caso de estudio se define la arquitectura mostrada en la Figura 2.

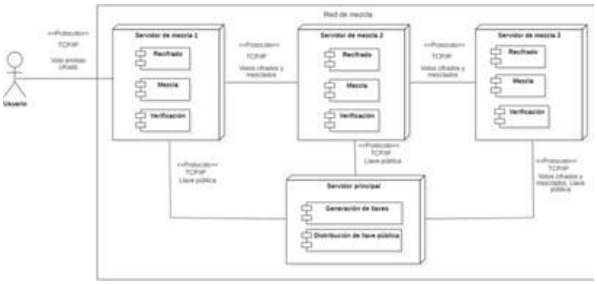

Figura 2. Arquitectura de red de mezcla [elaboración propia].

En la arquitectura definida se utilizan 4 servidores, con el objetivo de tener un mayor control con respecto al funcionamiento de los servidores de mezcla.

El servidor principal se encarga de generar las llaves pública y privada a utilizar en la red de mezcla, así como de compartir la llave pública con los servidores de mezcla.

El resto son los servidores de mezcla, los cuales se encargan de re-cifrar los votos recibidos, mezclarlos y verificar el proceso de mezcla.

El proceso de re-cifrado y mezcla genera una nueva lista de votos la cual se comparte con el siguiente servidor de mezcla.

Cada vez que se inicia uno de los servidores de mezcla, éste solicita la llave pública al servidor principal. Una vez que todos los servidores de mezcla obtienen la llave pública, los votos cifrados emitidos por los usuarios llegarán al servidor de mezcla 1, éste hará los procesos de mezcla y re-cifrado para enviar la nueva lista de votos al servidor de mezcla 2. El servidor de mezcla 2 repite los procesos de mezcla y re-cifrado para compartir su salida con el servidor de mezcla 3. Una vez que el servidor de mezcla 3 mezcla y re-cifra los votos, comparte su salida con el servidor principal, ésta es la lista final de votos obtenida.

Para la verificación del correcto mezclado nos basamos en el sistema de votación Helios. Este sistema utiliza el concepto de mezcla sombra que consiste en mezclar los votos cifrados usando una permutación independiente a la utilizada en la mezcla que se envió al siguiente servidor, así como utilizar factores de re-cifrado independientes. Un factor de re-cifrado es el número aleatorio utilizado en ElGamal para cifrar el número 1 que se multiplica en el proceso de re-cifrado. (Adida).

Al utilizar la misma lista de votos cifrados para la mezcla original y la mezcla sombra se provee integridad a los votos ya que si existe una modificación en los votos es posible conocer esta discrepancia e informar de que el mezclado no fue realizado correctamente.

Para un mejor entendimiento del proceso de verificación se explica de la siguiente manera:

El primer bloque M representa la mezcla original de la cual su salida (que son los votos permutados y re-cifrados) se transmite al siguiente servidor. El segundo bloque representa la mezcla sombra para la verificación, la cual como se mencionó utiliza factores de re-cifrado y permutaciones independientes a los empleados en la mezcla original.

Después se guardan la permutación y los factores de re-cifrado de la mezcla sombra, el servidor calcula con la salida (que son los votos permutados y re-cifrados) el hash, el cual es el valor por verificar.

El verificador que quiera asegurar un correcto mezclado en el servidor debe conocer la lista de votos cifrados usada, solicitar al servidor de mezcla la permutación y factores de re-cifrado, así como el hash. Con los valores anteriores es posible para el verificador calcular el hash

Para el cálculo del hash de verificación en el cual el verificador utiliza la lista de votos re-cifrados, la permutación y los factores de re-cifrado guardados para recrear la mezcla sombra. La salida obtenida que son los votos permutados y re-cifrados es usada para calcular su hash, al ser el mismo proceso que en el servidor de mezcla, el verificador compara los hashes obtenidos y al asegurar que son iguales se comprueba que el proceso de mezcla y re-cifrado fue exitoso.

Para la implementación de la red de mezcla con re-cifrado con verificación de mezcla se optó por el lenguaje de programación Python, aprovechando las implementaciones realizadas en el análisis de los algoritmos de ElGamal y de Paillier. Esto permitió aprovechar el código existente y acelerar el desarrollo de la red de mezcla.

Las funciones principales implementadas en el sistema fueron:

Mezclado y Re-cifrado de los votos: Estas funciones dan anonimato a los votos modificando el mensaje de entrada en un servidor de mezcla, en este caso los votos cifrados. Finalmente, el mensaje de salida del servidor es una representación diferente de los votos.

Verificación de correcto mezclado: A través de la verificación de correcto mezclado, se confirma que los servidores de mezcla han llevado a cabo de manera efectiva la mezcla y re-cifrado de los votos asegurando el anonimato de los votantes. Para este paso se utilizó la verificación del sistema de votación Helios a través del concepto de Mezclas Sombra. (Adida).

Interacción entre servidores: La comunicación entre los servidores implementa como se envían los votos cifrados, la llave pública que se utiliza en el proceso de cifrado y re- cifrado, así como los valores que se utilizan para verificar el correcto mezclado.

Los servidores se comunican por medio de sockets de flujo y los mensajes son enviados en formatos de cadenas codificadas a bytes.

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

El algoritmo que se utilizó para realizar la mezcla en los servidores se muestra en la Tabla 9, este proceso es el mismo en todos los servidores de mezcla, así como en el cálculo de la mezcla sombra.

Tabla 9. Algoritmo para generar la mezcla. [elaboración propia].

Algoritmo 9. Generar permutación
Entrada: Tamaño Arreglo de Votos Cifrados l
Salida: Permutación $perm$
Crear un arreglo secuencial $perm$ de tamaño l
Para i desde 0 hasta l hacer
Seleccionar un número aleatorio $d, tal\ que\ 0 \leq d \leq i$
Si i es diferente a d entonces
$perm[i] \leftarrow perm[d]$
$perm[i] \leftarrow i$
Devolver $perm$

Para la generación de la permutación se utilizó el algoritmo Fisher-Yates, en el cual partiendo del primer elemento de la lista se genera la permutación escogiendo un número aleatorio entre el 0 y el índice i, después se comprueba si d tiene el mismo valor de i, de ser así se intercambian el valor de la posición d con i. (Black, 2023)

En las pruebas realizadas para verificar el anonimato de los votos se muestra la siguiente, en la cual se muestran 3 votos iguales para el "Candidato1", del lado izquierdo se muestran los votos en texto plano y del lado derecho los votos en texto cifrado. Dado que el algoritmo de ElGamal es probabilístico, el texto cifrado de cada uno de los votos es distinto entre sí. Véase la Figura 3:

Figura 3. Votos generados que ingresan al servidor de mezcla 1 [elaboración propia].

A continuación, en la Figura 4 se muestra la lista resultante una vez que se procesan los votos en la red de mezcla 1, del lado izquierdo podemos ver que los votos de prueba persisten después del proceso de mezcla y re-cifrado, del lado derecho vemos que el texto cifrado es distinto para todos los votos, siendo esta la demostración de que el homomorfismo en la red de mezcla de re-cifrado.

Figura 4. Votos procesados por el servidor de mezcla 1 que ingresan al servidor de mezcla 2 [elaboración propia].

En la Figura 5 se muestran los votos una vez que son procesados por el servidor de mezcla 2, nuevamente vemos que los votos persisten y el re-cifrado de los votos funciona correctamente.

Figura 5. Votos procesados por el servidor de mezcla 2 que ingresan al servidor de mezcla 3 [elaboración propia].

Finalmente, en la figura 6 se muestran los votos una vez procesados por el servidor de mezcla 3, se puede apreciar que los votos persisten y el proceso de mezcla y re-cifrado funciona correctamente. Esta será la lista de votos final y la que llegará al servidor principal.

Figura 6. Votos procesados por el servidor de mezcla 3 que ingresan al servidor principal [elaboración propia].

II. Resultados

Con las pruebas realizadas en la red de mezcla implementada, se pudo validar el correcto funcionamiento de los procesos que

están en ella (Cifrado, Re-cifrado, Mezcla, Descifrado y Verificación).

Se valida el anonimato brindado a los votos con el uso del software desarrollado, de esta manera es viable implementar la red de mezcla desarrollada en un sistema de voto electrónico.

III. Conclusiones

Es necesario realizar la implementación de la red de mezcla en un lenguaje de programación más eficiente, dado al volumen de datos que se utilizan en una prueba real con 1000 votos se terminó el proceso en 30 minutos.

El uso de los algoritmos de cifrado homomórfico resultan una alternativa viable cuando se requiere tratar con datos sensibles, dado que permite manipular los datos sin necesidad de descifrar los mismos, se puede preservar la confidencialidad de la información en el tratamiento.

La arquitectura propuesta permite la implementación de más servidores de mezcla dentro de la red, aunque se debe trabajar en mejoras como la tolerancia a la inactividad de los servidores, así como la generación distribuida de llaves para evitar el uso del servidor principal.

Referencias

Adida, B. (n.d.). Helios: Web-based Open-Audit Voting. *17th USENIX Security Symposium.* Retrieved from https://www.usenix.org/legacy/events/sec08/tech/full_papers/adida/adida.pdf

Black, P. E. (2023, abril 5). *"Fisher-Yates shuffle".* Retrieved from National Institute of Standards and Technology: https://xlinux.nist.gov/dads/HTML/fisherYatesShuffle.html

Larsson, E., & Svensson, C. (n.d.). *El Gamal Mix-Nets and Implementation of a Verifier (Tésis de ingeniería) Facultad de Informática y comunicaciones.* Estocolmo, Suecia: KTH Royal Institute of Technology. Retrieved from https://www.diva-portal.org/smash/get/diva2:

Oujani, A. (2011). *Tools and Protocols for Anonymity on the Internet.* Retrieved from https://www.cse.wustl.edu/~jain/cse571-11/ftp/anonym/index.html

Pigatto, D. F. (2012). *Segurança em sistemas embarcados críticos - utilizaçao de criptografía para comunicaçao segura (Tesis doctoral, Universidad de São Paulo).* São Paulo. Retrieved from https://www.researchgate.net/publication/298421681_Seguranca_em_sistemas_embarcados_criticos_-_utilizacao_de_criptografia_para_comunicacao_segura

Ramchen, K. (2012). *Efficient mixnets with application to electronic voting (Tesis de maestría, Depto de Informática y Sistemas de Información.* Melbourne, Australia.