EL CIFRADO PAILLIER APLICADO AL VOTO ELECTRÓNICO

(The Paillier's cryptosystem applied to electronic voting)

Trabajo de Fin de Máster

para acceder al

MÁSTER EN MATEMÁTICAS Y COMPUTACIÓN

Autor: Lucas Bravo Villamil

Director: Daniel Sadornil Renedo

Septiembre – 2025

Índice general

Resumen

El Cifrado Paillier Aplicado al Voto Electrónico

Este trabajo estudia en profundidad el cifrado de Paillier y sus modificaciones, centrándose en su aplicación al voto electrónico. Tras introducir nociones de álgebra, criptografía y pruebas de conocimiento cero, se presenta el funcionamiento del criptosistema original, destacando sus propiedades homomórficas. Se analizan variantes como la de Damgård-Jurik-Nielsen, que mejora la capacidad de los mensajes y reduce el factor de expansión. Finalmente, se diseñan protocolos de votación electrónica seguros y verificables, que garantizan privacidad, integridad y recuento justo mediante descifrado con umbral.

The Paillier Cryptosystem Applied to Electronic Voting

This work provides an in-depth study of the Paillier cryptosystem and its modifications, focusing on its application to electronic voting. After introducing concepts from algebra, cryptography, and zero-knowledge proofs, the original cryptosystem is presented, highlighting its homomorphic properties. Variants such as the Damgård-Jurik-Nielsen scheme are analyzed, which improve message capacity and reduce the expansion factor. Finally, secure and verifiable electronic voting protocols are designed, ensuring privacy, integrity, and fair tallying through threshold decryption.

Palabras clave

- Cifrado de Paillier
- Voto electrónico
- Criptografía homomórfica
- Pruebas de conocimiento cero
- Descifrado umbral

Capítulo 1

Introducción

La democracia tiene su origen en la Atenas de la época clásica, en el V a.C. Los griegos desarrollaron este sistema de gobierno del pueblo para establecer un orden que se diferenciase de los de los demás imperios y polis de la Antigüedad, y que resultara más justo. La piedra angular de este sistema era la asamblea de ciudadanos en la que se tomaban las decisiones mediante votación por parte de los presentes, votación que se realizaba a mano alzada salvo en algunas excepciones. La votación se estableció como la forma más justa de toma de decisiones y a día de hoy sigue siendo el método que se utiliza. De cara al futuro, en las democracias modernas, el proceso de las votaciones ha tenido que ir cambiando. No se puede reunir a un país entero en un lugar y hacer una votación a mano alzada. Primero por su inviabilidad y segundo porque vulnera la privacidad de los votantes. Las urnas electorales fueron la solución que se encontró a este problema ya en época de los griegos. El voto se introducía de manera secreta en la urna y al finalizar se contaban todos sin saber a quien correspondía cada uno. Este método fue tan revolucionario que a día de hoy se sigue empleando y por el momento sin un candidato a sustituirlo. Las votaciones tienen cabida en muchos ámbitos de la vida, y a día de hoy, con la tecnología, estas se realizan también de forma telemática. El problema consiste en garantizar que una votación a través de internet sea segura, privada, íntegra y verificable.

Con estas necesidades presentes empezaron a surgir propuestas para votaciones electrónicas basadas en la criptografía moderna. En concreto, las propuestas más interesantes incluyen entre sus características ser un cifrado no determinista y homomórfico. La propiedad homomórfica permite realizar transformaciones en el texto cifrado que se transmiten al texto en claro original. En concreto se pueden operar textos cifrados y será como operar con los textos en claro. Esta propiedad se usará para el recuento. Esto facilita el recuento de los votos sin tener que descifrar cada uno de ellos de forma individual. Dentro de esta categoría de cifrados se encuentra el que nos ocupa en este trabajo, el cifrado de Paillier, publicado en 1999 por Pascal Paillier. El cifrado de Paillier es un cifrado homomórfico de clave pública no determinista. A lo largo del trabajo vamos a tratar la totalidad del criptosistema, la generación de claves, el cifrado y el descifrado, así como

un análisis de la seguridad del mismo.

A pesar de que puede parecer que el cifrado de Paillier es muy adecuado para la votación electrónica no está exento de problemas. Por ejemplo, el texto cifrado ocupa mucho en relación al texto en claro, el algoritmo de descifrado no es muy eficiente, las exponenciaciones son muy lentas... En este trabajo se abordan modificaciones al esquema de Paillier que tratan de lidiar con estos problemas, siendo la más importante la de Damgård, Jurik y Nielsen. Lo que pretenden con su modificación es, por una parte, aumentar la longitud de los mensajes que se pueden cifrar y, por otra, que el factor de expansión, es decir, la relación entre el texto cifrado y el original, se reduzca considerablemente hasta aproximarse a 1. Las otras variantes que comentaremos tienen menos relevancia: una fue propuesta por el propio Paillier en el mismo artículo en el que presentaba su cifrado y busca reducir el tiempo de descifrado. La otra fue propuesta por Catalano, Gennaro, Howgrave y Nguyen y buscaba mantener el esquema de Paillier pero reduciendo el exponente a uno más pequeño. De esta forma las operaciones serían más ágiles.

En la última sección del trabajo presentaremos el voto electrónico con sus elementos, fases y requisitos de seguridad. Principalmente, buscaremos mantener en todo momento la privacidad del voto, la integridad del mismo y garantizar un recuento justo de la votación. De cara a conseguir esto usaremos el cifrado de Damgård-Jurik-Nielsen(DJN_s) para diseñar protocolos de votación electrónica. Tenemos que conseguir que el esquema de votación sea lo más parecido posible a un sistema en el que todos los votantes y las autoridades sean honestos y no curiosos. En primer lugar veremos los esquemas de voto y recuento para diferentes tipos de votaciones con el cifrado DJN_s. A continuación, veremos todos los protocolos con los que verificar que un voto es válido en los diferentes tipos de votaciones. Se podrá verificar que el voto emitido es una de las opciones posibles y, en los casos de voto múltiple, que no está repetido. Por último hablaremos del descifrado con umbral, esta mecánica garantiza que un número de autoridades inferior a un cierto umbral no pueda descifrar ni votos ni el resultado de la votación. Esto último es esencial para garantizar que una autoridad corrupta no puede acceder a los votos de forma individual.

En definitiva, el objetivo del trabajo es conocer en profundidad el cifrado de Paillier y su modificación propuesta por Damgård, Jurik y Nielsen y aplicar estos cifrados al voto electrónico. Esto nos va a permitir crear esquemas de votación seguros y privados, pero que a su vez garanticen que la votación esté teniendo lugar de manera adecuada.

Capítulo 2

Preliminares

En este trabajo será muy necesario conocer algunos aspectos de aritmética modular, así como de números primos y algunas funciones especiales como la φ de Euler o la λ de Carmichael. Además necesitaremos conocer conceptos básicos de criptografía y en concreto de reparto de secretos. Por otro lado, en el trabajo se va a utilizar el problema de la mochila en uno de los esquema de voto de la sección 5 por lo que también se introduce aquí. Por último hablaremos de una herramienta muy útil para probar que se conoce una cierta información sin revelarla, las pruebas de conocimiento cero.

2.1. Álgebra

Las definiciones y conceptos más básicos se pueden consultar en cualquier libro de álgebra como el [5]. Empezamos la sección de preliminares con un concepto que será necesario para discutir posteriormente la función φ de Euler y la función λ de Carmichael.

Si tomamos los elementos de $\mathbb{Z}/\mathrm{n}\mathbb{Z}$ que tienen inverso multiplicativo, estos forman un grupo con la aplicación multiplicación de enteros módulo n. Este grupo se denota como $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y viene definido de la siguiente manera:
Definición 2.1. $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^* = \{z \in \mathbb{Z}/\mathrm{n}\mathbb{Z} \mid \text{ mcd}(z, n) = 1\}$

Con la identidad de Bezout se puede probar que un elemento que cumple esa condición tiene inverso multiplicativo. En particular, si n es un número primo p sucede que $(\mathbb{Z}/p\mathbb{Z})^*$ tiene $p-1$ elementos ya que todos los elementos de $\mathbb{Z}/p\mathbb{Z}$ tienen inverso excepto el 0. Es obvio por la definición anterior ya que $\text{mcd}(p, a) = 1$ para todo $0 \neq a \in \mathbb{Z}/p\mathbb{Z}$. Entonces, no solo tenemos que forman parte del grupo multiplicativo todos los elementos no nulos de $\mathbb{Z}/p\mathbb{Z}$, sino también que $\mathbb{Z}/p\mathbb{Z}$ es un cuerpo con la suma y la multiplicación.
Definición 2.2. La función φ de Euler: $\varphi : \mathbb{N} \longrightarrow \mathbb{N}$ asigna a cada n un valor que es el número de elementos primos con n y menores que n. Es decir: $\varphi(n) = \#\{a \in \mathbb{Z}/\mathrm{n}\mathbb{Z} \mid \gcd(a, n) = 1\}$

Es inmediato por la definición de la función que:

$$\varphi(n) = \left|(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*\right| \tag{2.1}$$

Más adelante necesitaremos tomar elementos en grupos de esta forma concreta y este resultado nos permite saber cuántos elementos tienen dichos grupos. En particular es interesante conocer esta función en los números primos y en los compuestos que serán los que necesitemos más adelante.

Antes de ver la proposición correspondiente a esto vamos a presentar uno de los teoremas más importantes del álgebra, que es:

Teorema 2.1 (Teorema chino de los restos)**.** *Sea* $\{n_i\}_{i=1}^k$ *un conjunto de números naturales cumpliendo que:* mcd$(n_i, n_j) = 1\ \forall i \neq j$*. Y sea* $n = \prod_{i=1}^k n_i$*. Entonces el sistema lineal de congruencias:*

$$\begin{cases} x \equiv a_1 \bmod n_1 \\ x \equiv a_2 \bmod n_2 \\ \qquad \vdots \\ x \equiv a_k \bmod n_k \end{cases}$$

Tiene una única solución módulo n.

Demostración: La prueba del teorema chino de los restos tiene dos partes: en la primera vamos a construir la solución y en la segunda vamos a probar que dicha solución es única.

En primer lugar vamos a considerar $\tilde{n}_j = \prod_{i=1, i\neq j}^k n_i$, es decir el producto de todos los números menos el j-ésimo. Como se cumple que para cada par $i \neq j$ mcd$(n_i, n_j) = 1$ se tiene entonces que mcd$(\tilde{n}_j, n_j) = 1$. Usando la identidad de Bezout podemos afirmar que existen x_j y t_j tales que: $1 = \tilde{n}_j x_j + n_j t_j$. Por tanto x_j es el inverso multiplicativo de $\tilde{n_j}$ módulo n_j y además $\tilde{n_j} x_j = 1 \bmod n_i$ para todo $i \neq j$.

A partir de esto es muy sencillo construir la solución del sistema, sea $x = \sum_{i=1}^k \tilde{n}_i x_i a_i$, por construcción es claro que x satisface el sistema de ecuaciones del enunciado.

La unicidad la demostraremos mediante una reducción al absurdo. Es decir, supongamos que existe x' otra solución del sistema distinta. Como tanto x como x' son soluciones satisfacen todas las ecuaciones del sistema: $x \equiv a_j \equiv x' \bmod n_j$ luego $x - x' \equiv 0 \bmod n_j$. Es decir que $n_j | x - x'$ para todo j. Por tanto, como los n_i son primos entre sí, $n = \prod_{i=1}^k n_i | x - x'$ y entonces ya tenemos la unicidad módulo n probada porque $x - x' \equiv 0 \bmod n$ lo que implica que $x \equiv x' \bmod n$. □

El TCR puede entenderse también como un homomorfismo de grupos que se muestra a continuación:

Corolario 2.1. *Sea* $\{n_i\}_{i=1}^k$ *un conjunto de números naturales cumpliendo que para cada* $i \neq j$ mcd$(n_i, n_j) = 1$ *para todo* $i \neq j$*. Y sea* $n = \prod_{i=1}^k n_i$*. Se cumple que las aplicaciones:*

$$\begin{array}{rccc} \mu: & \mathbb{Z}/\mathrm{n}\mathbb{Z} & \longrightarrow & \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z} \\ & x \bmod n & \hookrightarrow & (x \bmod n_1,\ x \bmod n_2, \ldots,\ x \bmod \tilde{n}) \end{array}$$

$$\begin{array}{rccl} \mu^* : & (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* & \longrightarrow & (\mathbb{Z}/n_1\,\mathbb{Z})^* \times (\mathbb{Z}/n_2\,\mathbb{Z})^* \times \cdots \times (\mathbb{Z}/n_k\,\mathbb{Z})^* \\ & x \bmod n & \hookrightarrow & (x \bmod n_1,\ x \bmod n_2, \ldots,\ x \bmod \tilde{n}) \end{array}$$

Son isomorfismos.

Ahora sí podemos ver la proposición sobre la función φ de Euler en primos y compuestos.
Proposición 2.1. *Sea p un número primo. Entonces se cumple que:*

$$\varphi(p) = p - 1$$

Sean n y ñ dos números naturales tales que mcd$(n, \tilde{n}) = 1$*. Entonces se cumple que:*

$$\varphi(n\tilde{n}) = \varphi(n)\varphi(\tilde{n})$$

Demostración: Lo primero es obvio porque $|(\mathbb{Z}/p\,\mathbb{Z})^*| = p - 1$. Para demostrar la otra propiedad de la función φ de Euler vamos a recurrir a la propia definición de la misma y vamos a trabajar con los grupos multiplicativos y sus cardinales. Por tanto, la proposición es cierta si y solo si se satisface lo siguiente: $(\mathbb{Z}/n\tilde{n}\,\mathbb{Z})^* \simeq (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* \times (\mathbb{Z}/\tilde{n}\,\mathbb{Z})^*$

Definimos una aplicación de la siguiente manera:

$$\begin{array}{rccl} \mu : & (\mathbb{Z}/n\tilde{n}\,\mathbb{Z})^* & \longrightarrow & (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* \times (\mathbb{Z}/\tilde{n}\,\mathbb{Z})^* \\ & x \bmod n\tilde{n} & \hookrightarrow & (x \bmod n,\ x \bmod \tilde{n}) \end{array}$$

Directamente del corolario del teorema chino de los restos 2.1 tenemos que esa aplicación es un isomorfismo. Y de este isomorfismo podemos deducir el valor de la función φ de Euler en $\tilde{n}n$ cuando son primos entre sí. □

El caso particular que nos va a interesar a nosotros es para p y q primos distintos y $n = pq$:

$$\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1) \tag{2.2}$$

Siguiendo con propiedades de la función φ de Euler observamos que:

Sea p un primo y k un número natural, entonces se tiene que:

$$\varphi(p^k) = p^{k-1}(p-1) = p^k\left(1 - \frac{1}{p}\right) \tag{2.3}$$

Sea $n = p_1^{e1}p_2^{e2}...p_k^{ek}$ entonces se tiene que:

$$\varphi(n) = n\prod_{i=1}^{k}\left(1 - \frac{1}{p_i}\right) \tag{2.4}$$

Además se puede ver que el grupo $(\mathbb{Z}/p^n\,\mathbb{Z})^*$ es cíclico para todo p impar y para todo

$n > 0$ [10]. Es ya sabido que $\mathbb{Z}/p\mathbb{Z}$ es cíclico. Antes del teorema vamos a probar un lema técnico:

Lema 2.1. *Sea p primo impar. para todo y y $n > 1$ $y^p \equiv 1 \mod p^{n+1}$ si y solo si $y \equiv 1 \mod p^n$*

Demostración: Partamos de $y \equiv 1 \mod p^n$, entonces $y = 1 + kp^n$ para algún $k \in \mathbb{Z}$. Si elevamos a p usando la fórmula del binomio de Newton nos quedaría $y^p \equiv 1 + kp^{n+1} \mod p^{n+2}$. Como podemos observar entonces $y \equiv 1 \mod p^n$ implica $y^p \equiv 1 \mod p^{n+1}$.

Ahora que sabemos que la implicación hacia la izquierda se cumple vamos con la otra por inducción. En el caso $n = 1$ tenemos que $y^p \equiv 1 \mod p^2$. Entonces también es cierto que $y^p \equiv 1 \mod p$, si $y^p - 1$ es divisible entre p^2, en particular también lo es entre p. Además $y^p \equiv y \mod p$ ya que por el pequeño teorema de Fermat $y^{p-1} \equiv 1 \mod p$, por tanto $1 \equiv y^p \equiv y \mod p$.

Ahora vamos a poner $n \geq 1$ y ver que si $y^p - 1$ es divisible por p^{n+1} entonces $y - 1$ es divisible por p^n, supongamos que se cumpla para n y vamos a probarlo para $n + 1$. Supongamos que $y^p - 1$ es divisible entre p^{n+2}, entonces es divisible entre p^{n+1} también. Entonces tenemos que $y-1$ es divisible entre p^n, es decir que $y = 1+kp^n$. Si lo elevamos a p tenemos que $y^p = 1+pkp^n+(p(p-1))k^2p^{2n}+\cdots \equiv 1+kp^{n+1} \mod p^{n+2}$. Pero habíamos dicho que y^p-1 era divisible entre p^{n+2}, por lo tanto kp^{n+1} también, luego $p|k$. por lo tanto $y - 1 = kp^n$ es divisible por p^{n+1}. □

Se puede ver que el grupo $(\mathbb{Z}/p^n\mathbb{Z})^*$ es cíclico para todo p impar y para todo $n > 0$ [10] usando el lema anterior y que $(\mathbb{Z}/p\mathbb{Z})^*$ es cíclico.

Teorema 2.2 (Teorema de Gauss)**.** *Sea p un primo impar. Entonces para todo $n > 0$, $(\mathbb{Z}/p^n\mathbb{Z})^*$ es cíclico.*

Demostración: Sea x un generador de $(\mathbb{Z}/p\mathbb{Z})^*$, es decir un elemento de orden $p - 1$. Primero veamos que o x o $x + p$ tienen orden $p(p - 1)$ en $(\mathbb{Z}/p^2\mathbb{Z})^*$ y por lo tanto es cíclico.

En primer lugar sabemos que el orden en $(\mathbb{Z}/p^2\mathbb{Z})^*$ de x, sea a, tiene que dividir al cardinal del grupo, por tanto $a|p(p - 1)$. Pero además sabemos que $x^a \equiv 1 \mod p$ de forma obvia, por tanto $(p - 1)|a$. Por tanto, $a = p - 1$ o $a = p(p - 1)$. Si es el segundo caso ya estaría. Si es el primero se hace la misma discusión para $x + p$ y se llega a que su orden es $p(p - 1)$ o $(p - 1)$, en el primer caso acabamos. Solo falta ver que a la vez no se pueden dar ambos segundos casos simultáneamente. Supongamos que el orden de x es $p - 1$:

$$(x + p)^{p-1} \equiv x^{p-1} + (p - 1)x^{p-2}p \equiv 1 - px^{p-2} \not\equiv 1 \mod p^2$$

entonces $x + p$ tiene que tener $p(p - 1)$.

A continuación probaremos para $n \geq 2$ que si tenemos un generador z de $(\mathbb{Z}/p^n\mathbb{Z})^*$ entonces es un generador de $(\mathbb{Z}/p^{n+1}\mathbb{Z})^*$. Como hemos visto que hay un generador $z = x \ o \ x + p$ cuando $n = 2$, el primer paso de la inducción está cubierto, una vez

probado esto tendremos el resultado.

Supongamos que z tiene orden $|(\mathbb{Z}/p^n\,\mathbb{Z})^*| = p^{n-1}(p-1)$ en $(\mathbb{Z}/p^n\,\mathbb{Z})^*$. Con una discusión similar a la del párrafo anterior se puede probar que si el orden de z en $(\mathbb{Z}/p^{n+1}\,\mathbb{Z})^*$ es $p^n(p-1)$, y se terminaría el resultado, o $p^{n-1}(p-1)$. Sin embargo, en el segundo caso, podemos aplicar el lema 2.1 con $y = z^{p^{n-2}(p-1)}$, de forma que como se cumple $y^p \equiv 1 \mod p^{n+1}$, se tiene que también $y \equiv 1 \mod p^n$, lo cual es absurdo con la suposición que hemos hecho sobre el orden de z. El segundo caso es imposible y el teorema está probado. □

Otro resultado importante del grupo de unidades es el teorema de Euler.

Teorema 2.3 (Teorema de Euler)**.** *Sean a y n* $\in \mathbb{Z}$ *con gcd(a,n) = 1, entonces se cumple que:*

$$a^{\varphi(n)} \equiv 1 \mod n$$

Demostración: Si a es un elemento que cumple mcd$(a, n) = 1$ entonces significa que $a \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$. Como $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ es un grupo con la multiplicación podemos usar propiedades de los grupos. En concreto que el orden de los elementos de un grupo divide al orden del grupo, y por tanto t = orden(a)$|(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*|$. Es decir que existe un $k \in \mathbb{Z}$ tal que $tk = |\,(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*\,| = \varphi(n)$ y se tiene que:

$$a^{\varphi(n)} \equiv a^{tk} \equiv (a^t)^k \equiv 1^k \equiv 1 \mod n \qquad \square$$

Otra función relacionada con $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ es la función de Carmichael que determina el máximo orden de los elementos de un grupo y se define a continuación.

Definición 2.3. Se define la función λ de Carmichael como $\lambda(n) : \mathbb{N} \longrightarrow \mathbb{N}$ tal que $\lambda(n) = min\left\{k | a^k \equiv 1 \mod n \ \forall a \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*\right\}$

Estas dos funciones tienen la misma propiedad de que al elevar un elemento de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ a su valor da 1. Sin embargo, φ no está definida con esta propiedad, simplemente es un valor para el que se cumple siempre que $a^{\varphi(n)} \equiv 1 \mod n$ para todo $a \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ porque coincide con el orden del grupo de unidades 2.3. Como consecuencia directa del teorema y la definición de λ tenemos que para todo n se cumple que $\lambda(n)|\varphi(n)$. En particular estas van a coincidir en los grupos en los que haya un elemento de orden máximo, es decir, en los grupos cíclicos. El ejemplo más claro se tiene cuando n es primo:

Ejemplo 2.1. $\varphi(p) = (p-1) = \lambda(p)$

Ejemplo 2.2. $\varphi(8) = 4$ frente a $\lambda(8) = 2$. En este caso tenemos que $(\mathbb{Z}/8\,\mathbb{Z})^* = \{1,\ 3,\ 5\ \ 7\}$. Cualquiera de estos números al cuadrado verifican que son congruentes con 1 módulo 8. Evidentemente a la cuarta también lo cumplirán.

Proposición 2.2. *Sean n y ñ dos enteros que cumplen que gcd(n,ñ) = 1. Entonces:*

$$\lambda(n\tilde{\mathrm{n}}) = \mathrm{mcm}\,(\lambda(n),\ \lambda(\tilde{\mathrm{n}}))$$

Demostración: Sea la aplicación $\mu : (\mathbb{Z}/n\tilde{n}\,\mathbb{Z})^* \longrightarrow (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* \times (\mathbb{Z}/\tilde{n}\,\mathbb{Z})^*$ definida por

$\mu(x \bmod n\tilde{n}) \hookrightarrow (x \bmod n,\ x \bmod \tilde{n})$ que sabemos que es un isomorfismo por el teorema chino de los restos (Corolario 2.1). Entonces se tiene que cumplir que $\mu(1) = (1,\ 1)$. Por lo tanto, si tenemos $a^k \equiv 1 \bmod n\tilde{n}$ con $a \in (\mathbb{Z}/n\tilde{n}\,\mathbb{Z})^*$ se tiene que cumplir que:

$$\mu(a^k) = \mu(1) = (1,\ 1) = (a^k \bmod n,\ a^k \bmod \tilde{n})$$

Por lo tanto, de la definición de la función $\lambda(n)$ tenemos que $\lambda(n)|k$ y $\lambda(\tilde{n})|k$, luego como eso se va a cumplir para todos los k que hagan $a^k \equiv 1 \bmod n\tilde{n}$ tenemos que el valor más pequeño posible es mcm $(\lambda(n),\ \lambda(\tilde{n})) = \lambda(n\tilde{n})$ □

En el caso particular de dos primos distintos p y q tales que $n = pq$ (que es el que nos va interesar a nosotros) tendríamos que:

$$\lambda(n) = \text{mcm}(\lambda(p),\ \lambda(q)) = \text{mcm}((p-1),\ (q-1)) \tag{2.5}$$

Teorema 2.4 (Teorema de Carmichael)**.** *Sean p y q dos primos impares distintos, sea $n = pq$ y sea $s \in \mathbb{N}$. Entonces, para cada $a \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ se cumple que:*

$$a^{n^s\lambda(n)} \equiv 1 \bmod n^{s+1}$$

Demostración: Partimos de un elemento $a \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$, en esta situación tenemos que se cumple:

$$a^{n^s\lambda(n)} \equiv \left(a^{\varphi(p^{s+1})}\right)^{q^s\frac{\lambda(n)}{\lambda(p)}} \equiv 1\ (mod\, p^{s+1})$$
$$a^{n^s\lambda(n)} \equiv \left(a^{\varphi(q^{s+1})}\right)^{p^s\frac{\lambda(n)}{\lambda(q)}} \equiv 1\ (mod\, q^{s+1})$$

Para ver la primera equivalencia tenemos que recordar (2.3) para ver que $\varphi(p^{s+1}) = p^s(p-1)$. Además de que $\lambda(p) = (p-1)$. Entonces es claro sustituyendo que:

$$n^s\lambda(n) = p^sq^s\lambda(n) = p^s\frac{(p-1)}{(p-1)}q^s\lambda(n) = \varphi(p^{s+1})q^s\frac{\lambda(n)}{\lambda(p)}$$

Y análogamente se cumple para la otra congruencia. La segunda parte es clara por el teorema de Euler 2.3 y la misma congruencia se tiene módulo q^{s+1}

Para concluir la demostración vamos a usar el teorema chino de los restos 2.1. Notamos que estamos en las condiciones necesarias para aplicarlo: mcd$(p^{s+1},\ q^{s+1}) = 1$ y se obtiene que:

$$a^{n^s\lambda(n)} \equiv 1\ (mod\, n^{s+1})$$

□

2.2. Criptografía

La criptografía es una disciplina que hace referencia a un amplio rango de temas en la seguridad a la hora de transmitir información. A lo largo de la historia siempre ha existido la necesidad de una comunicación segura entre individuos, bien por guerras, dinero,

orgullo, influencias... piezas sensibles de información que si se filtrasen a la persona equivocada podrían desembocar en una catástrofe. La primera aparición de un cifrado data de la Antigua Grecia. Los griegos cifraban sus mensajes con un instrumento llamado escítala. Más adelante en el siglo I a.C. Julio César desarrolló otro de los cifrados más antiguos y conocido de la historia, en su honor se le conoce como cifrado César. Mucho tiempo después, con una idea muy parecida, pero un poco mejorada, aparece el cifrado de Vignère. Desde entonces los criptosistemas han ido evolucionando constantemente, sin embargo, su necesidad sigue siendo la misma, garantizar una comunicación segura entre individuos. Consultaremos las definiciones y conceptos básicos de la criptografía en [8].

Un sistema criptográfico está formado por $\mathcal{M}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{E}$ y $\mathcal{D}$. $\mathcal{M}$ representa al conjunto de mensajes en claro m, son los mensajes originales antes de ser cifrados. $\mathcal{C}$ representa al conjunto de posibles mensajes cifrados c. $\mathcal{K}$ es el conjunto de posibles claves. $\mathcal{E}$ es una función de cifrado que va de $\mathcal{M} \times \mathcal{K}$ en $\mathcal{C}$. $\mathcal{D}$ es una función de descifrado que va de $\mathcal{C} \times \mathcal{K}$ en $\mathcal{M}$. Para un mensaje m, una clave de cifrar $eK \in \mathcal{K}$ y una clave de descifrar $dK \in \mathcal{K}$ se tiene que cumplir que $\mathcal{D}(\mathcal{E}(m, eK), dK) = m$ para todo $m \in \mathcal{M}$.

Un cifrado tiene que ser "$seguro$", entendiéndose seguro en el sentido de que la aplicación que se usa para cifrar no sea fácil de obtener observando simplemente el texto cifrado. Aún así, si se conoce la función de cifrado podemos seguir garantizando la seguridad del cifrado si esta es de una sola vía.

Definición 2.4. Función de una sola vía: se dice que una función $f : X \longrightarrow Y$ es de una sola vía cuando es sencillo calcular $f(x) = y \; \forall x \in X$, pero es difícil calcular $f^{-1}(y) = x$ para casi cualquier $y \in Im(f) \subset Y$.

Este tipo de funciones son más bien útiles para procesos de verificación. Para la comunicación se utilizó durante muchos años el cifrado de clave privada:

Definición 2.5. Criptosistema de clave privada: se dice que un criptosistema es de clave privada cuando para cifrar y descifrar se usa la misma clave o una puede obtenerse de la otra y esta es conocida con anterioridad por los interlocutores de la comunicación.

Estos sistemas funcionaron bien durante muchos años en los ámbitos en los que era necesario cifrar mensajes. Sin embargo, para utilizar un criptosistema de clave privada es necesario haber acordado antes una clave por un canal seguro. En la década de los '70 aparece el primer sistema de clave pública:

Definición 2.6. Función trampa: es una función $f : X \longrightarrow Y$ con la cual es fácil calcular $f(x)$ para un elemento $x \in X$ si se tiene cierta información que llamaremos la clave pública y que es de una sola vía, excepto si se conoce una porción de información extra que permite invertirla. A esta la llamaremos clave privada.

Se dice que un criptosistema es de clave pública cuando usa una función trampa. En un criptosistema de clave pública la información de cifrado está a disposición de cualquier persona mientras que la necesaria para descifrarlo se mantiene en secreto. Supongamos que tenemos dos individuos Alicia y Berto, y estamos usando un sistema de clave pública. Berto quiere mandarle un mensaje a Alicia, para ello va a mirar la clave pública

de Alicia en su directorio público y va a cifrar el mensaje. Ahora se lo manda a Alicia y esta lo puede descifrar usando su clave privada porque le permite invertir la función de trampa. Si un adversario interceptara el mensaje cifrado, este no podría descifrarlo supuestamente porque sin la clave privada de Alicia la función es de una sola vía. Este paradigma de criptofrafía pública o asimétrica tiene la ventaja de que no es necesario acordar una clave para la comunicación por adelantado. Es utilizado en el envío de mensajes confidenciales, en autenticación, en intercambio de claves, sorteos aleatorios (*coin flip*), distribución de secretos, pruebas de conocimiento cero, de las cuales hablaremos en la sección 2.4.

Los cifrados de clave pública deterministas tienen bastantes problemas para probar su seguridad en ataques de texto en claro conocido. En este tipo de ataques el adversario tiene acceso a algo que le permite cifrar los textos elegidos por él y ver cómo quedan. En concreto en el contexto de una votación en la que el número de textos en claro es tan limitado, el cifrado de clave pública más famoso, RSA, que es determinista, flaquea en cuanto a seguridad. Si el cifrado es determinista significa que un texto cifrado siempre se cifra de la misma manera, por tanto un adversario simplemente tiene que cifrar todos los posibles textos y comparar. Por eso son mucho más interesantes lo cifrados de clave pública probabilísticos como el que tratamos en este trabajo.

2.2.1. Reparto de secretos

A veces es necesario guardar una información de carácter confidencial, pero de una forma que pueda ser accesible por un conjunto de personas autorizadas. Para ello, una de las estrategias que se realizan es el reparto de secretos [1], por ejemplo, el código de lanzamiento de un misil se reparte entre los oficiales del ejército y hacen falta por lo menos n de ellos para lanzar dicho misil.

Definición 2.7. El reparto de secretos consiste en crear unas particiones a partir de un secreto, ya sean creadas a partir de él o con él, y estas se reparten entre un grupo de usuarios de forma que ciertos subconjuntos de los mismos puedan recuperar la información y otros no.

Supongamos que tenemos unos participantes $\mathscr{P} = \{P_1, \ldots, P_n\}$, entonces una estructura de acceso consiste en un conjunto $\Gamma \subset P(\mathscr{P})$ cuyos elementos son las denominadas agrupaciones autorizadas. $\mathscr{K}$ es el secreto y $\mathcal{S}$ es el conjunto de participaciones a repartir. Cuando se desee obtener el secreto $k \in \mathscr{K}$ a partir de un esquema, el gestor D repartirá a cada participante $P_i \in \mathscr{P}$ una participación $s_i \in S$ mediante unas reglas de distribución. Idealmente de forma que un subconjunto autorizado pueda recuperar el secreto y un subconjunto no autorizado no.

Esquema de Shamir

Fue el primero de los esquemas de reparto de secretos [13], propuesto por Ami Shamir en 1979. La idea es sencilla en origen, utiliza el teorema de existencia y unicidad de

interpolación polinómica, que dice que existe un único polinomio de grado d que pasa por $d+1$ puntos. Por tanto lo que se va a hacer es calcular un polinomio aleatorio de grado $d-1$ sobre $K[x]$ siendo K un cuerpo y con la restricción de que $f(0) = s$ que es el secreto que queremos repartir. Una vez se tiene esto se calculara el valor de la función en n puntos y se repartirán entre los participantes, para estandarizar el proceso digamos que a cada participante P_i se le entrega $f(x_i = i) := y_i$, de forma que el punto al que tiene acceso es al (i, y_i). De esta forma d autoridades cada una con su participación y_i pueden reconstruir el polinomio $f(x)$ y obtener la clave privada.

Una forma ágil de calcular el polinomio usando $H = d$ autoridades es usando los polinomios de Lagrange:

$$L_{H,i}(x) = \prod_{j=1 \wedge j \neq i}^{H} \frac{x - x_j}{x_i - x_j}$$

Estos polinomios tienen la propiedad de valer 0 en todos los puntos del soporte distintos del i-ésimo y 1 en este. Se tiene por tanto que el valor del polinomio es:

$$f(x) = \sum_{i=1}^{H} y_i L_{H,i}$$

Así que ya tenemos el polinomio que necesitamos, pero como lo queremos es su ordenada en el origen podemos simplemente hacer:

$$s = f(0) = \sum_{i=1}^{H} y_i \prod_{j=1 \wedge j \neq i}^{H} \frac{-x_j}{x_i - x_j} = \sum_{i=1}^{H} y_i L_{H,i}(0) \tag{2.6}$$

De esta forma d participantes pueden obtener el secreto s y un número inferior a este de los mismos no. La estructura de acceso Γ contiene los subconjuntos de cardinal superior o igual d.

2.3. Optimización: el problema de la mochila

En optimización combinatoria existe un problema muy famoso conocido como el problema de la mochila que consiste en buscar la mejor manera de llenar una mochila cuya capacidad es limitada [7]. Con "buscar la mejor manera" entramos en cuál es la función a optimizar. En el caso más genérico cada objeto constará de un valor y un peso, el objetivo del problema será introducir en la mochila la combinación de objetos cuyos pesos sumen menos que el máximo P y que maximicen la suma de los valores de los mismos. Dado un objeto i del conjunto total de objetos O, tenemos que su peso es p_i y su valor es v_i. El objetivo es maximizar $\sum_{i \in I} v_i$ con $I \subset O$ de forma que $\sum_{i \in I} p_i < P$.

Si existe más de un objeto disponible de cada clase el problema es un poco más complicado porque la función a maximizar es $\sum_{i \in I} n_i v_i$ y la restricción es $\sum_{i \in I} n_i p_i < P$. En

nuestro caso, lo que nos va a interesar en el trabajo es una versión de este problema en el que no interviene el valor de los objetos. El problema consiste en minimizar $\sum_{i \in I} n_i$ de forma que se cumpla $\sum_{i \in I} n_i p_i < P$. Es decir, llenar la mochila con el menor número de objetos posible. Si los pesos son supercrecientes o potencias de un entero resolverlo es fácil, pero en general no tiene porque ser así.

2.4. Pruebas de conocimiento cero

Para ser realistas con los protocolos de votación y crear un método que funcione de verdad en el mundo real es imprescindible tener en cuenta la falta de honestidad. Por tanto, en los sistemas de voto se utilizan lo que se conoce como *pruebas de conocimiento*, formas que tienen tanto los votantes como las autoridades de demostrar que siguieron correctamente el protocolo. Para el caso de los votantes es demostrar que han votado por alguna de las posibles opciones sin desvelar por cual, para garantizar la robustez respetando la privacidad. En concreto ese tipo de pruebas se conocen como *pruebas de conocimiento cero* o *Zero-knowledge proof*(ZKP).

Definimos los siguientes conceptos:
Definición 2.8. Un protocolo de prueba interactiva es una forma de conversación entre un individuo P que es el que demuestra y V que es el verificador. P afirma conocer un secreto y el objetivo es convencer a V de la veracidad de esta afirmación. V puede aceptar o rechazar la prueba.
Definición 2.9. Se denomina prueba de conocimiento a una prueba interactiva, es decir una conversación entre las partes, si cumple que:

- Completitud: Si P y V son honestos se acepta la prueba con probabilidad 1.
- Solvencia: Si P no es honesto entonces un V honesto tiene una probabilidad despreciable de aceptar la prueba. Si V acepta la prueba de P con probabilidad no despreciable, entonces existe un algoritmo polinómico capaz de extraer el secreto de P y por tanto P no podría estar mintiendo.

Definición 2.10. Una prueba de conocimiento cero, PCC, es una prueba de conocimiento en la que si V fuera deshonesto no obtendría nada de información sobre P más allá de que es honesto, no podría conocer el secreto de P.
Ejemplo 2.3. Supongamos que Pablo quiere probarle a su amigo Víctor que conoce la contraseña para abrir una puerta interior de una cueva que conecta las entradas derecha e izquierda, pero no le quiere revelar la contraseña porque tiene intención de vendérsela. Para ello idean una prueba de conocimiento cero: Pablo va a entrar por una de las entradas de forma aleatoria, una vez dentro le va a pedir a Víctor que grite por qué puerta quiere que salga, en función de la respuesta Pablo saldrá o atravesará la puerta y saldrá por la otra entrada según le diga Víctor. Como Pablo se mete en la cueva por una entrada aleatoria si no conociera la contraseña hay una probabilidad de 0.5 de que Víctor diga la misma entrada y Pablo salga por ella. Si esta prueba se repite un número suficiente de

veces la probabilidad de que Pablo salga bien todas las veces sin conocer la contraseña es despreciable, por lo que si sucede, Víctor puede afirmar casi seguro que Pablo tiene la contraseña sin necesidad de que Pablo se la revele.

Se puede probar que un protocolo es PCC si existe un simulador en tiempo polinómico S, capaz de simular una conversación válida de la prueba con solamente el conocimiento que tienen en común P y V y ningún acceso a P y la distribución de probabilidad es la misma que la de la prueba entre P y V. En concreto nos van a interesar los Σ-protocolos.

Definición 2.11. Se dice que una prueba tiene solvencia especial si para cualquier x y cualquier par de conversaciones aceptadas (a, e, z) y $(a, \tilde{\mathrm{e}}, \tilde{\mathrm{z}})$, con $e \neq \tilde{\mathrm{e}}$, calcular el secreto ω es factible.

Definición 2.12. Un σ-protocolo es un protocolo en el que se parte de un input común entre P y V, llamémoslo x, que suele ser el input de un problema computacional, (en nuestro caso es el c, n y s) y P conoce un secreto de x, el input privado llamado *testigo*, ω de x (en nuestro caso será la $r \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$). A continuación se realizan las siguientes tres comunicaciones entre P y V:

1. $P \to V$: P crea una variable aleatoria a y un compromiso hacia ella, es decir que luego no la podrá cambiar, y se la envía a V.
2. $V \to P$: V reta a P con un número de longitud t bits aleatorio e.
3. $P \to V$: P responde z, que será el resultado de operar de cierta manera que depende del tipo de prueba con a y con e así como el secreto ω que P quiere demostrar que conoce.

V decide si acepta o no en función de la conversación (a, e, z) y x. Además, para ser σ-protocolo tiene que cumplir la completitud, la solvencia especial.

Definición 2.13. Una prueba se dice que es de conocimiento cero con verificador honesto, PCCV_H, si existe un simulador S que con un input x genera una conversación (a, e, z) que tiene una distribución de probabilidad como la conversación real entre P y V sobre x.

Definición 2.14. Una prueba de conocimiento cero se dice que es de verificador honesto especial, PCCV_HE, si existe un simulador S que con inputs x y e puede crear una conversación aceptada (a, e, z). Esta tiene que tener la misma distribución de probabilidad que la conversación real entre P y V sobre x en la que V reta a P con el número e.

Estas definiciones quedarán claras más adelante con los ejemplos de las pruebas que vamos a necesitar para el sistema de votación.

Definición 2.15. Un Σ-protocolo es un σ-protocolo que además es PCCV_HE.

Necesitamos estos Σ-protocolos aunque no sean exactamente PCC porque si en el papel del verificador colocamos una función aleatoria el sistema se puede volver no interactivo y además PCC, todo junto sería una prueba de conocimiento cero no interactiva, PCC_NI. Estas son muy interesantes en la práctica, pero a nivel teórico son solo una adaptación de los Σ-protocolos, así que no vamos a entrar en ellas.

Capítulo 3

Cifrado Paillier

El cifrado Paillier apareció por primera vez en un artículo del matemático francés Pascal Paillier de 1999, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes* [11]. Este cifrado de clave pública comparte algunos aspectos con el famoso RSA que se utiliza aún hoy en día como cifrado estándar en algunas comunicaciones. Para empezar, ambos son de clave pública y si se rompe el RSA se puede romper el cifrado de Paillier, como veremos más adelante. Mientras que uno basa su seguridad en la imposibilidad por parte de un tercer usuario de factorizar un número compuesto suficientemente grande (el RSA), el otro basa su seguridad en la imposibilidad de calcular el valor de la función de Carmichael para un número compuesto lo suficientemente grande. Además, el cifrado propuesto por Paillier tiene una serie de propiedades homomórficas que serán las que nos interesarán de cara a aplicar este cifrado a un recuento de votos. Cuando usamos este cifrado se puede operar con los textos cifrados sin descifrarlos y el texto en claro se verá modificado de forma acorde a la transformación. [12]

En primer lugar vamos a describir el algoritmo de generación de claves del criptosistema planteado por Paillier, así como el cifrado y el descifrado.

3.1. Generación de claves

En primer lugar vamos a presentar el algoritmo de generación de claves. Este algoritmo lo primero que hace es buscar un primo grande, entiéndase por grande un número primo de cientos de dígitos, y luego otro de forma que obtengamos una pareja de primos que al multiplicarlos se obtenga un número compuesto admisible.

Definición 3.1. Un número n se dice que es *admisible* si cumple que $n = pq$ con p y q primos distintos y mcd$(n, \varphi(n)) = 1$

Dado n dicho número compuesto admisible se tomará d como $\lambda(n) = d$, esto es lo que será la clave privada y la seguridad del criptosistema está basada en la intratabilidad de la factorización de n. Pues sin ella el valor de la función de Carmichael $\lambda(n)$ es prácticamente imposible de obtener (2.5). Lo último que necesitamos como parte de la clave es un elemento aleatorio de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$. La elección aleatoria de este elemento no es

necesaria ya que no ofrece más seguridad, si fijásemos g a un elemento cualquiera de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ cuyo orden fuera un múltiplo no nulo de n el criptosistema sería igualmente seguro, como veremos más adelante.

Algorithm 1 Algoritmo $\mathcal{K}_P$

$\mathcal{K}_P()$:
$p \leftarrow$ primo grande
$q \leftarrow$ primo grande
while $\gcd(pq, \varphi(pq)) \neq 1$ **do**
    $q \leftarrow$ primo grande
end while
$n \leftarrow pq$
$d \leftarrow \lambda(n)$
$g \leftarrow 2$
while $\text{orden}_{n^2}(g) \neq 0 \mod n$ **do**
    $g \leftarrow$ elemento aleatorio de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$
end while
$eK \leftarrow (n, g)$
$dK \leftarrow (d)$
Salida: $\leftarrow (eK,\ dK)$

3.2. Cifrado

El algoritmo de cifrado es muy sencillo, lo primero que hace es elegir un elemento aleatorio $r \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$, y luego calcula el mensaje cifrado c mediante la siguiente aplicación.

$$\begin{array}{rccc} \gamma_g: & \mathbb{Z}/\mathrm{n}\mathbb{Z} \times (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* & \longrightarrow & \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^* \\ & (m,\ r) & \hookrightarrow & g^m r^n \end{array} \tag{3.1}$$

Algorithm 2 Algoritmo $\mathcal{E}_P$

$\mathcal{E}_P(m, eK)$:
$r \leftarrow$ elemento aleatorio de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$
$c \leftarrow \gamma_g(m,\ r) = g^m r^n \mod n^2$
Salida: $\leftarrow c \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$

Esta aplicación en realidad es biyectiva entre ambos grupos, pero solo es isomorfismo cuando el orden de g es exactamente n.

Para ver que es biyectiva lo primero que haremos será fijarnos en que la cardinalidad de ambos conjuntos coincide.

$$\left|\mathbb{Z}/\mathrm{n}\mathbb{Z} \times (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*\right| = |\mathbb{Z}/\mathrm{n}\mathbb{Z}| \cdot \left|(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*\right| = n \cdot \varphi(n) = n(p-1)(q-1) = \varphi(n^2) = \left|\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*\right|$$

Luego la biyectividad se puede verificar simplemente comprobando la inyectividad.

Para ello tomaremos dos elementos $(m_1,\ r_1)$ y $(m_2,\ r_2)$ que cumplen:

$$g^{m_1} r_1^n \equiv g^{m_2} r_2^n \text{ mód } n^2$$

Como $r_1 \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ se tiene que mcd$(r_1, n) = 1$ y por tanto mcd$(r_1, n^2) = 1$, entonces $r_1 \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ luego también tiene inverso módulo n^2 y podemos despejar:

$$g^{m_2 - m_1} \left(\frac{r_2}{r_1}\right)^n \equiv 1 \text{ mód } n^2 \tag{3.2}$$

Elevamos a $\lambda(n)$ en ambos lados:

$$g^{(m_2 - m_1)\lambda(n)} \left(\frac{r_2}{r_1}\right)^{n\lambda(n)} \equiv 1 \text{ mód } n^2$$

Usando el teorema de Carmichael 2.4:

$$g^{(m_2 - m_1)\lambda(n)} \equiv 1 \text{ mód } n^2$$

Por tanto $(m_2 - m_1)\lambda(n)$ es un múltiplo del orden de g, que sabemos que es un múltiplo de n luego $(m_2 - m_1)\lambda(n) \equiv 0 \text{ mód } kn$ y se tiene que $(m_2 - m_1)\lambda(n) \equiv 0 \text{ mód } n$

Como mcd$(n,\ \lambda(n)) = 1$ es claro que debe ocurrir $m_2 \equiv m_1 \text{ mód } n$ sustituyendo en (3.2) se tiene que: $r_2^n \equiv r_1^n \text{ mód } n^2$. Como mcd$(n,\ \lambda(n)) = 1$ usando la identidad de Bezout se tiene que existen $u,\ v \in \mathbb{Z}$ tales que $1 = nu + \lambda(n)v$, es decir $nu = 1 - \lambda(n)v$. Si elevamos a u en ambos lados podemos escribir $r_2^{1-\lambda(n)v} \equiv r_1^{1-\lambda(n)v} \text{ mód } n^2$ es decir $r_2 r_2^{-\lambda(n)v} \equiv r_1 r_1^{-\lambda(n)v} \text{ mód } n^2$

Esta congruencia también es cierta módulo n por ser un n factor y teniendo en cuenta la definición de $\lambda(n)$ se concluye:

$$r_2 \equiv r_1 \text{ mód } n$$

Luego queda probada la inyectividad y con ella la biyectividad.

Faltaría verificar que se conserva la operación del grupo por la aplicación para verificar que es un homomorfismo y por tanto un isomorfismo. Esta propiedad afirmamos que se tiene para $g \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ de forma que su orden sea exactamente n, elemento que existe porque $n|\varphi(n^2) = \left|\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*\right| = n(p-1)(q-1)$. Entonces ahora verificamos que se cumple:

$$\gamma_g(m_1 + m_2,\ r_1 r_2) = \gamma_g(m_1,\ r_1)\gamma_g(m_2,\ r_2)$$

Es necesario tener en cuenta que la suma y el producto de los argumentos se tiene módulo n y la multiplicación en la imagen es módulo n^2

$$\gamma_g(m_1 + m_2 \text{ mód } n,\ r_1 r_2 \text{ mód } n) = g^{m_1 + m_2 \text{ mód } n}(r_1 r_2 \text{ mód } n)^n \text{ mód } n^2$$

Entonces para la g adecuada nos queda: $g^{m_1+m_2 \text{ mód } n} = g^{m_1+m_2} \text{ mód } n^2$ pues $m_1 + m_2 =$

$kn + \delta$ entonces $m_1 + m_2 \bmod n = \delta$. En el otro término de la equivalencia podemos verificar mediante una simple sustitución que $g^{kn+\delta} \equiv g^{kn} g^{\delta} \equiv g^{\delta} \bmod n^2$ y por tanto se cumple.

Por otra parte, sabemos que $r_1 r_2 \bmod n = r_1 r_2 + kn$ para algún $k \in \mathbb{Z}$ y:

$$(r_1 r_2 + kn)^n \equiv \sum_{j=0}^{n} \binom{n}{j} (r_1 r_2)^{n-k} (kn)^k \equiv (r_1 r_2)^n + n(r_1 r_2)^{n-1} kn + ... \equiv (r_1 r_2)^n \bmod n^2$$

Juntando ambas igualdades:

$$\begin{aligned} \gamma_g(m_1 + m_2 \bmod n,\ r_1 r_2 \bmod n) &\equiv g^{m_1+m_2 \bmod n} (r_1 r_2 \bmod n)^n \bmod n^2 \\ &\equiv g^{m_1+m_2} (r_1 r_2)^n \bmod n^2 \\ &\equiv g^{m_1} r_1^n g^{m_2} r_2^n \bmod n^2 \\ &\equiv \gamma_g(m_1, r_1)\gamma_g(m_2, r_2) \bmod n^2 \end{aligned}$$

Por tanto hemos verificado que la aplicación es un homomorfismo de grupos cuando orden(g)=n en $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$.

Por lo tanto queda probado que γ_g es un isomorfismo entre ambos grupos.

La elección aleatoria de $r \in \left(\mathbb{Z}/\mathrm{n}\mathbb{Z}\right)^*$ provoca que haya un total de $\varphi(n)$ posibles cifrados del mismo mensaje m por (2.1). No va a suceder que si ciframos el mismo mensaje m con r_1 y r_2, siendo $r_1 \neq r_2$ obtengamos el mismo mensaje cifrado c. Esto es muy importante para su aplicación al voto electrónico. Que la aplicación que se usa para cifrar sea un homomorfismo es otra característica que en un futuro le dará su aplicabilidad al voto electrónico. $\gamma_g(m, r)$ será nuestro mensaje cifrado que es un elemento de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$, que es el grupo de unidades de $\mathbb{Z}/\mathrm{n}^2\mathbb{Z}$ con la operación producto. El mensaje $m \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$, que es un grupo con la suma, cuando realicemos multiplicaciones entre mensajes cifrados estos resultarán como cifrar la suma de los mensajes en claro con la clave pública correspondiente. Un criptosistema con esta propiedad se denomina homomórfico. Esto será relevante más adelante en el recuento de votos.

Ahora vamos a entrar en la discusión sobre el elemento g de la generación de claves del cifrado de Paillier y porque la podemos fijar sin renunciar a la seguridad. Lo primero que vamos a ver es el concepto de residuo n-ésimo.

Definición 3.2. Decimos que un elemento $z \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ es un residuo n-ésimo si existe un elemento $y \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ tal que:

$$z \equiv y^n \bmod n^2$$

Estos elementos forman un subgrupo de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ de $\varphi(n)$ elementos. Para ver esto definiremos el conjunto $R_{PA} = \left\{ r^n |\ r \in \left(\mathbb{Z}/\mathrm{n}\mathbb{Z}\right)^* \right\}$ que se puede ver de forma inmediata que es un subgrupo de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$. Una vez tenemos el subgrupo, para justificar el cardinal del mismo vamos a usar la aplicación potencia n-ésima $\mu : \left(\mathbb{Z}/\mathrm{n}\mathbb{Z}\right)^* \to R_{PA}$ definida

como $\mu(r) = r^n \bmod n$. Las propiedades de homomorfismo se heredan directamente de γ_g y la suprayectividad es obvia por la definición de R_{PA}. Para ver la inyectividad tomamos $r_1^n \equiv r_2^n \bmod n$ y de nuevo podemos ver que el razonamiento usado en γ_g es válido aquí porque nuestro n es admisible. Teniendo que μ es un isomorfismo ya queda claro que R_{PA}, que son los residuos n-ésimos de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$, forman un subgrupo de $\varphi(n)$ elementos.

De hecho cada residuo n-ésimo módulo n^2 se puede probar que es congruente a $a^n \bmod n^2$ para un único a con $0 < a < n$. En primer lugar es sabido que $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^* = \langle g \rangle$ pues es un grupo cíclico, luego cada elemento del grupo $y \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ se puede escribir como $y \equiv g^k \bmod n^2$ para algún k. Si x es un residuo n-ésimo entonces existe un y que cumple $y^n \equiv x \bmod n^2$, entonces existe k tal que, $y^n \equiv (g^k)^n \equiv g^{kn} \equiv x \bmod n^2$. Entonces todos los residuos n-ésimos son de la forma g^{tn} para algún entero t. Ahora lo que tenemos que encontrar es un a tal que $a^n \equiv g^{tn} \bmod n^2$ y $0 < a < n$. Si g es una raíz primitiva módulo n^2, también lo es módulo n, si g es primo con n^2 también lo es con n. Además para cada t existe un único $a \in \{1, 2, \ldots, n-1\}$ tal que $a \equiv g^t \bmod n$. Por el teorema del levantamiento de Hensel [15] si $a \equiv g^t \bmod n$, entonces $a^n \equiv (g^t)^n \bmod n^2$ para un a apropiado. Falta verificar la unicidad, supongamos que $a_1^n \equiv x \bmod n^2$ y $a_2^n \equiv x \bmod n^2$ con $0 < a_1, a_2 < n$, entonces se tiene que $a_1 \equiv a_2 \bmod n$ porque mcd$(n, \varphi(n)) = 1$. La limitación de a_1 y a_2 fuerza que $a_1 = a_2$.

Decidir si un elemento de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ es un residuo n-ésimo es un problema aleatoriamente auto-reducible (habitualmente conocido por sus siglas en inglés RSR (*random self-reducible*)). Significa que hay un algoritmo en tiempo polinómico que reduce una instancia del problema a otra aleatoria del mismo problema. Entonces si conseguimos un algoritmo que funcione bien para algún caso o subconjunto de casos, podemos resolver todas las instancias del problema porque podemos reducir los otros casos a uno de estos en un tiempo polinómico. Esto implica que el problema o es intratable para todas las instancias o por el contrario es polinómico para todas las instancias. La seguridad del criptosistema de Paillier se apoya en que el problema de decidir si un elemento es un residuo n-ésimo módulo n^2 es un problema intratable, más adelante veremos por qué.

De ahora en adelante llamaremos al problema de decidir si un elemento es un residuo n-ésimo o no como CR[n] y a la hipótesis de que este problema es intratable como DCRA (*Decisional Composite Rediduosity Assumption*).

Definición 3.3. Dado $g \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ tal que $orden(g) = 0 \bmod n$. Entonces para un $w \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ llamamos clase residual n-ésima de w con respecto a g al único entero $x \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$ tal que existe un $r \in \left(\mathbb{Z}/\mathrm{n}\mathbb{Z}\right)^*$ tal que $\gamma_g(x, r) = w$.

$$[\![w]\!]_g = x$$

Esta definición tiene sentido, porque como ya hemos visto γ_g es una biyección, por tanto para cada elemento $w \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ existe un único elemento de $\mathbb{Z}/\mathrm{n}\mathbb{Z} \times \left(\mathbb{Z}/\mathrm{n}\mathbb{Z}\right)^*$ tal que $\gamma_g(x, r) = w$. La clase residual n-ésima se correspondería con el elemento del espacio $\mathbb{Z}/\mathrm{n}\mathbb{Z}$, es decir el mensaje. Por tanto todos los posibles cifrados de un mismo mensaje

nos dará que tienen la misma clase sobre ese g. En particular de esto se puede deducir que:

$$[\![w]\!]_g = 0 \Leftrightarrow w \text{ es un residuo } n\text{-ésimo módulo } n^2 \tag{3.3}$$

Lo cual tiene sentido porque si la clase es 0 significa que en el cifrado solo aparece la correspondiente r elevada a n luego es un residuo n-ésimo y como estas son exactamente $\varphi(n)$, que se corresponde con el número de residuos n-ésimos, todos los residuos de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ cumplen que su clase es 0. Al problema de calcular la clase residual n-ésima de un elemento en base g la denotaremos por $Class[n, g]$, es decir a calcular $[\![w]\!]_g$ a partir solamente de w. Este problema al igual que el de decidir si un elemento es un residuo n-ésimo es aleatoriamente auto-reducible(RSR) sobre g.

Lema 3.1. *El problema Class[n,g] es RSR sobre g. Es decir, dados $g_1, g_2 \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ de orden un múltiplo no nulo de n:*

$$Class[n, g_1] \equiv Class[n, g_2]$$

Demostración: Lo primero que vamos a observar en la demostración es que:

$$[\![w]\!]_{g_1} = [\![w]\!]_{g_2} [\![g_2]\!]_{g_1} \text{ mód } n$$

Esto se puede comprobar escribiendo las expresiones de lo que es w tanto tomando γ_{g_1} como γ_{g_2}. Es decir los elementos $(x_1, r_1),\ (x_2, r_2) \in \mathbb{Z}/\mathrm{n}\mathbb{Z} \times (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ que cumplen $\gamma_{g_1}(x_1, r_1) = w$ y $\gamma_{g_2}(x_2, r_2) = w$. Como $g_2 \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ puedo encontrar un par de elementos $(x_3, r_3) \in \mathbb{Z}/\mathrm{n}\mathbb{Z} \times (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ tales que $\gamma_{g_1}(x_3, r_3) = g_2$. Es decir, $[\![w]\!]_{g_1} = x_1$, $[\![w]\!]_{g_2} = x_2$ y $[\![g_2]\!]_{g_1} = x_3$.

De las igualdades $w = g_2^{x_2} r_2^n$ y $g_2 = g_1^{x_3} r_3^n$ se tiene que $w = g_1^{x_2 x_3} (r_3^{x_2} r_2)^n$. Como $w = g_1^{x_1} r_1^n$ se cumple que $x_1 = x_2 \cdot x_3 \text{ mód } n$, lo que es equivalente a decir que $[\![w]\!]_{g_1} = [\![w]\!]_{g_2} [\![g_2]\!]_{g_1} \text{ mód } n$.

De la misma manera podemos ver que $[\![w]\!]_{g_2} = [\![w]\!]_{g_1} [\![g_1]\!]_{g_2} \text{ mód } n$. Por lo tanto vemos que $[\![g_1]\!]_{g_2} = [\![g_2]\!]_{g_1}^{-1} \text{ mód } n$ y en consecuencia tenemos que $[\![g_2]\!]_{g_1}$ es invertible módulo n. Ahora recordemos que queríamos ver la equivalencia entre los problemas. Supongamos que tenemos un método de determinar el problema $Class[n, g_1]$, entonces podríamos solucionar el problema $Class[n, g_2]$ simplemente tomando:

$$[\![w]\!]_{g_2} = [\![w]\!]_{g_1} [\![g_2]\!]_{g_1}^{-1} \text{ mód } n$$

□

Es decir, gracias a este lema podemos asegurar que la complejidad solo depende de n y no del elemento tomado. Por tanto, podemos denotar $Class[n]$ al problema de calcular la base residual compuesta.

Como la complejidad del problema no depende de g, se suele utilizar una versión más sencilla del cifrado de Paillier en la que g no es un elemento cualquiera de orden adecuado de la clave pública, sino que se puede fijar siempre en un valor de $(n+1)$. Veamos

qué es una g válida.

Proposición 3.1. *Para todo entero $a \in \mathbb{N}$ se cumple que:*

$$(n+1)^a \equiv 1 + an \bmod n^2$$

Y además, el orden de $(n+1)$ módulo n^2 es n.

Demostración: Del desarrollo del binomio de Newton se tiene que: $(n+1)^a \equiv 1 + an \bmod n^2$.

Para calcular el orden es suficiente con observar que para todo entero a positivo menor que n el elemento an no se anula nunca y para n se cumple que $(n+1)^n \equiv 1 \bmod n^2$. □

Entonces $g = (n+1)$ no solo nos vale, sino que además hace que la aplicación $\gamma_{(n+1)}$, de ahora en adelante denotada γ por simplicidad, sea un isomorfismo biyectivo. Recordemos que la aplicación era biyectiva siempre, pero solamente era un homomorfismo de grupos si $orden(g) = n$.

3.3. Descifrado

Solo queda ver el algoritmo de descifrado para recuperar el mensaje a partir del texto cifrado. Para definir dicho algortimo en primer lugar vamos a introducir la función L:

$$\begin{array}{rccc} L: & S_n & \longrightarrow & \mathbb{Z}/\mathrm{n}\mathbb{Z} \\ & u & \hookrightarrow & \frac{u-1}{n} \end{array}$$

Siendo $S_n := \{u | 1 + kn \text{ con } 0 \leq k \leq n-1\}$, es decir es una función que va de los elementos de $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ que son 1 módulo n a $\mathbb{Z}/\mathrm{n}\mathbb{Z}$. De esta forma $L(1+kn) = k$. Si recordamos el teorema de Carmichael 2.4 podemos ver fácilmente que la función se puede aplicar a cualquier mensaje cifrado c, pues $c^d \equiv c^{\lambda(n)} \equiv 1 \bmod n$, luego $c^d \in S_n$.

Algorithm 3 Algoritmo $\mathcal{D}_P$

$\mathcal{D}_{Ps}(c, dK = d)$:
$N \leftarrow L(c^d \bmod n^2) = \frac{(c^d \bmod n^2)-1}{n}$
$D \leftarrow L(g^d \bmod n^2) = \frac{(g^d \bmod n^2)-1}{n}$
$m \leftarrow \frac{N}{D} \bmod n$
Salida: $\leftarrow m \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$

Para ver que el algoritmo recupera el mismo mensaje que se cifró vamos a usar la versión en la que $g = (n+1)$. En este caso, para descifrar el mensaje c basta con calcular:

$$\frac{L(c^d \bmod n^2)}{L(g^d \bmod n^2)} = \frac{L(c^{\lambda(n)} \bmod n^2)}{L((n+1)^{\lambda(n)} \bmod n^2)} \bmod n$$

En primer lugar vamos a desarrollar el numerador. Como: $c \equiv (n+1)^m r^n \bmod n^2$, al elevarlo a la función de Carmichael lo que obtendremos será: $c^{\lambda(n)} \equiv (n+1)^{m\lambda(n)} r^{n\lambda(n)} \bmod$

n^2. Por el teorema de Carmichael: $c^{\lambda(n)} \equiv (n+1)^{m\lambda(n)} \bmod n^2$ ($r^{n\lambda(n)} \equiv 1 \bmod n^2$). A partir de la proposición 3.1 se tiene que $c^{\lambda(n)} \equiv 1 + m\lambda(n)n \bmod n^2$. De la misma forma el denominador: $(n+1)^{\lambda(n)} \equiv 1 + n\lambda(n) \bmod n^2$. Ahora $L((n+1)^{\lambda(n)} \bmod n^2) = \frac{(n+1)^{\lambda(n)} \bmod n^2 - 1}{n} = \frac{1+n\lambda(n)-1}{n} = \lambda(n) = d$. Luego:

$$\frac{L(c^{\lambda(n)} \bmod n^2)}{L((n+1)^{\lambda(n)} \bmod n^2)} \equiv \frac{c^{\lambda(n)} \bmod n^2 - 1}{nd} \equiv \frac{1 + mdn - 1}{nd} = m \bmod n$$

En el caso general, para un valor cualquiera de g de orden múltiplo de n vamos a usar que todo elemento $w \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ se puede escribir como $w = (n+1)^x r^n$ (pues γ_g es un isomorfismo) con x y r adecuados. De la definición 3.3 se tiene que $[\![w]\!]_{n+1} = x$:

$$L(w^{\lambda(n)} \bmod n^2) = \frac{(1+n)^{x\lambda(n)} r^{n\lambda(n)} - 1}{n} = \frac{1 + x\lambda(n)n - 1}{n} = \lambda(n)x = \lambda(n)[\![w]\!]_{n+1} \bmod n$$

Tomamos el mensaje cifrado $c \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ y sabemos que $c = g^m r^n = (n+1)^{[\![c]\!]_{n+1}} r_1^n$, ya que la función γ_g es una biyección para todo g pues $orden(g) = 0 \bmod n$. Por tanto, sustituyendo en la expresión del algoritmo de descifrado tanto c como $g = (n+1)^{[\![g]\!]_{n+1}} r_2^n$:

$$\frac{L(c^d \bmod n^2)}{L(g^d \bmod n^2)} = \frac{\lambda(n)[\![c]\!]_{n+1}}{\lambda(n)[\![g]\!]_{n+1}} = [\![c]\!]_{n+1}[\![g]\!]_{n+1}^{-1} = [\![c]\!]_g = m \bmod n$$

Como hemos podido comprobar la forma de recuperar el mensaje original funciona.

3.4. Seguridad y equivalencia con otros problemas

En la presente sección vamos a presentar qué problemas matemáticos complejos nos garantizan la seguridad del cifrado. Además, vamos a ver cómo si se rompe uno más complejo que el que garantiza el secreto, este también se puede considerar como roto y con él la seguridad del criptosistema.

Definición 3.4. Denotaremos por Fact[n] al problema de factorizar n siendo de la forma $n = pq$ con p y q primos distintos y grandes.

Definición 3.5. Denotaremos por RSA[n,e] al problema de calcular las raíces e-ésimas módulo n de un número. Siendo un número $n = pq$ de factorización desconocida.

Se sabe que ambos problemas son *difíciles* o *intratables*, como bien dice Paillier en su artículo [11].

Entonces podemos enunciar y demostrar equivalencias entre los problemas:

Proposición 3.2. *Si se dispone de un algoritmo que resuelve Fact[n], entonces se tiene un algoritmo que resuelve Class[n].*

Demostración: Como ya hemos visto en el algoritmo de descifrado si tenemos el valor de $\lambda(n)$ se puede resolver el problema Class[n] y como el cálculo de $\lambda(n)$ es muy sencillo si se conoce la factorización de n si se resuelve Fact[n] se puede resolver Class[n]. □

Proposición 3.3. *Si se dispone de un algoritmo que resuelve RSA[n,n], entonces se*

tiene un algoritmo que resuelve Class[n].

Demostración: Supongamos que tenemos una forma para resolver RSA[n,n], es decir a partir de un número módulo n se podría determinar su raíz n-ésima.

Partamos de un $w \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ del cual queremos conocer su $[\![w]\!]_{n+1}$, ya que como es aleatoriamente auto-reducible se puede calcular para $g = (n+1)$ y ser válido para cualquiera. Entonces, supongamos que tenemos $w \equiv (n+1)^x y^n \mod n^2$. Por la proposición 3.1 podemos deducir que $w \equiv y^n \mod n$. Luego y es la solución de RSA[n, n]. Esta y es la r aleatoria que tendríamos en el cifrado Paillier y de esta forma se tiene:

$$\frac{w}{y^n} \equiv (n+1)^x \equiv 1 + xn \mod n^2$$

Por tanto, $L(\frac{w}{y^n}) = x$ es la solución al problema Class[n] □

Definición 3.6. Denotaremos por CR[n] al problema de decisión de distinguir residuos n-ésimos en $\mathbb{Z}/\mathrm{n}^2\mathbb{Z}$ de los que no lo son.

En [11] Paillier conjetura que este problema es intratable, al igual que con los anteriores no está probada su intratabilidad, pero a día de hoy se puede asumir que es intratable pues tampoco se ha probado lo contrario. Aunque antes se probó que los residuos n-ésimos son de la forma $a^n \mod n^2$ con a un elemento de $\mathbb{Z}/\mathrm{n}\mathbb{Z}$, esta es la única información de la que se dispone y habría que probar con todos los elementos para decir que un elemento no lo es.

Definición 3.7. Denotaremos por D-Class[n] al problema de decisión asociado a Class[n]. Es decir, dado un $z \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$, g cuyo orden sea un múltiplo de n y $x \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$ saber si se cumple $[\![z]\!]_g = x$.

Proposición 3.4. *Los problemas CR[n] y D-Class[n] son equivalentes en el sentido que resolver uno de ellos permite resolver el otro. Además, si se dispone un procedimiento para resolver Class[n] se tiene también un algoritmo para resolver CR[n] o D-Class[n].*

Demostración: Es sencillo ver que Class[n] resuelve el problema de decisión D-Class[n], pues es más fácil verificar una solución que resolver el problema, por tanto esta parte es inmediata. Veamos ahora la equivalencia entre CR y D-Class:

Si tenemos una forma de resolver CR[n] se puede resolver D-Class[n]. Tomemos $z \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$, g válido y $x \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$ y veamos como se puede decidir con CR[n]. Supongamos que $z = g^{x'} r^n$, en principio $x' \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$ es el elemento de $\mathbb{Z}/\mathrm{n}\mathbb{Z}$ que cumple que $\gamma_g(x', r) = z$ y queremos ver si es x. En primer lugar plantearemos $zg^{-x} \equiv g^{x'-x} r^n \mod n^2$. Por tanto, si ahora resolvemos CR[n] para $zg^{-x} \mod n^2$ por (3.3) este nos dirá que sí cuando $x' - x \equiv 0 \mod n$. Entonces la respuesta de D-Class[n] será sí cuando la respuesta de CR[n] sea sí y será no cuando sea no.

Por otra parte si tenemos una forma de resolver D-Class[n] podemos resolver también CR[n]. Para ello elegimos un elemento $z \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$. Si queremos ver si es un residuo n-ésimo tenemos que fijarnos en que en el esquema de Paillier sería como si fuera uno de los posibles cifrados de un mensaje $m = 0$ (3.3), pues entonces sería la potencia n-

ésima de un elemento de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ módulo n^2. Debido a que son los posibles cifrados del mensaje $m = 0$, se tiene que $z = g^0 r^n$ y por tanto da igual que g se use en el problema de decisión D-Class. Solamente tenemos que resolver $(z, g, 0)$, si la respuesta de D-Class es sí la de CR[n] también y si es negativa la de CR[n] también. □

Recopilando estos resultados podemos escribir la jerarquía de problemas:

$$CR[n] \equiv D - Class[n] \Leftarrow Class[n] \Leftarrow RSA[n] \Leftarrow Fact[n]$$

Recordemos que RSA[n] necesita que no se conozca la factorización de n, entonces en cuanto se puede factorizar n ya hemos roto RSA[n]. Como podemos ver los problemas Class[n] que son los que necesitan ser seguros para el cifrado de Paillier son intratables por ambos lados, tanto el problema en sí como el de decisión.

Capítulo 4

Modificaciones del cifrado Paillier

El cifrado de Paillier según fue propuesto por Paillier en 1999 tiene una serie de buenas propiedades que ya hemos visto en el capítulo anterior. Nos permite cifrar un mensaje m menor que n en un texto cifrado c menor que n^2. De primeras esto tiene dos limitaciones clave: el mensaje no puede ser más largo que n y además, tanto m como c ocupan una cantidad de bits diferente y muy dispar que depende de la elección del n, concretamente c ocupa el doble que m. En la modificación del cifrado propuesta por Ivan Damgård, Mads Jurik y Jesper Buus Nielsen en [4] se propone una forma de aumentar la longitud del texto en claro y el texto cifrado incluso cuando la clave pública ya ha sido fijada. Lo interesante de su modificación es que no se pierde la propiedad homomórfica del mismo y tampoco se pierde en seguridad, además se reduce notablemente el factor de expansión, la relación entre los tamaños del texto cifrado y el texto original.

4.1. Cifrado de Damgård-Jurik-Nielsen

Mientras que el cifrado de Paillier se basa en una computación en $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ lo que proponen Damgård, Jurik y Nielsen [4] es hacer que el cifrado sea en $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ con $s \in \mathbb{N}$. Como se puede aumentar el espacio del mensaje cogiendo una s más grande hace desaparecer gran parte de las limitaciones de espacio del esquema original, sin necesidad de buscar primos más grandes.

4.1.1. Nuevo esquema criptográfico

Lo primero que vamos a notar es una serie de propiedades del grupo multiplicativo $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$. Como vimos en el capitulo 2 el cardinal de este grupo es $\varphi(n^{s+1}) = n^s(p-1)(q-1) = n^s\varphi(n)$. En el caso de un n admisible tenemos que $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^* = G \times H$ donde G es un grupo cíclico de orden n^s, es decir isomorfo a $\mathbb{Z}/\mathrm{n}^{\mathrm{s}}\mathbb{Z}$, y H es un grupo isomorfo a $\left(\mathbb{Z}/\mathrm{n}\mathbb{Z}\right)^*$. Este hecho se puede deducir fácilmente del teorema chino de los restos 2.1 y del teorema de Gauss 2.2. El grupo cociente $\overline{G} = \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^* / H$ también es un cíclico de orden n^s.

Para llevar cierto paralelismo con el capítulo 3 vamos a recordar que la propiedad homomórfica del cifrado solo se mantenía cuando el elemento g era de orden exactamente n en $\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$. En nuestro nuevo contexto vamos a necesitar elementos que sean de orden n^s en $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$. Parece intuitivo probar en primer lugar si el valor de $g = n + 1$ tomado en el caso original cumple la condición.

Lema 4.1. *Para un n admisible y $s < p, q$ el elemento $(n + 1)$ tiene orden n^s en $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$.*

Demostración: Vamos a empezar considerando $(n + 1)^i = \sum_{j=0}^{i} \binom{i}{j} n^j$. Se tiene que $(n + 1)^i \equiv 1 \bmod n^{s+1}$ si y solo si $\sum_{j=1}^{s} \binom{i}{j} n^{j-1} \equiv 0 \bmod n^s$. Esto es claro simplemente despejando de $\sum_{j=0}^{i} \binom{i}{j} n^j = 1 + \sum_{j=1}^{i} \binom{i}{j} n^j \equiv 1 \bmod n^{s+1}$ ya que implica que $n^{s+1} | \sum_{j=1}^{i} \binom{i}{j} n^j$, por tanto dividiendo entre n en ambos lados $n^s | \sum_{j=1}^{i} \binom{i}{j} n^{j-1}$.

Esta condición se cumple si $i = n^s$ pues si $j > s$ entonces el término que acompaña al número combinatorio es obvio que es divisible por n^s. En caso contrario como $s < p, q$ entonces $j!$ no pueden contener a p ni a q como factores primos y por tanto no divide al n^s del numerador y sobrevive como factor de forma íntegra. Por tanto podemos afirmar que el orden de $(n + 1)$ divide a n^s. Es decir es de la forma $p^\alpha q^\beta$ con $\alpha, \beta \leq s$. Fijemos $a = p^\alpha q^\beta$ y vamos a considerar términos $\binom{a}{j} n^{j-1}$ de la suma $\sum_{j=1}^{a} \binom{a}{j} n^{j-1}$. Veamos que todo término de esa suma es divisible entre a. Esto es trivial si $j > s$, y para $j \leq s$, se tiene porque $j!$ no puede tener a p o a q como factores primos, (tenemos que $s < p, q$), entonces $a | \binom{a}{j}$. Ahora vamos a suponer por reducción al absurdo que $a < n^s$. Sin pérdida de generalidad podemos suponer que $\alpha < s$. Sabemos que n^s divide a $\sum_{j=1}^{a} \binom{a}{j} n^{j-1}$ y también que a divide a ambos. Nos queda entonces que p tiene que dividir $\sum_{j=1}^{a} \binom{a}{j} n^{j-1}/a$. Sin embargo, el primer término de esta suma es 1 y aunque el resto si son divisibles entre p tenemos que $\sum_{j=1}^{a} \binom{a}{j} n^{j-1}/a \equiv 1 \bmod p$, lo cual nos lleva a contradicción. De esta forma concluimos que el orden de $(n + 1)$ en $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ es n^s. □

Como el orden de H es $\varphi(n)$ que es primo con n^s ya que n era un número admisible y eso implica que mcd$(n, \varphi(n)) = 1$. Por tanto tenemos que $\overline{(n + 1)} = (1 + n)H \in \overline{G}$ es un generador de $\overline{G}$, excepto quizás si $s > p, q$. Si en un grupo existe un elemento con el mismo orden de este el grupo es cíclico y es generado por él. Además es necesario que el orden de H sea primo con el de $\overline{G}$ para asegurar que el elemento $(n+1)$ no pertenezca a H ni ninguna de sus potencias tampoco, de forma que mantenga el orden en el grupo cociente. Entonces las clases a izquierda del subgrupo H en $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ son:

$$H, (1 + n)H, (1 + n)^2 H, \ldots, (1 + n)^{n^s - 1} H$$

Lo cual nos numera de forma natural todos las clases laterales, que serán todos los elementos de $\overline{G}$. Vamos a estudiar más en profundidad la estructura de $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ que permite dar una generalización natural del criptosistema de Paillier.

Lema 4.2. *Para un n admisible y $s < p, q$, la aplicación $\psi_s : \mathbb{Z}/\mathrm{n}^{\mathrm{s}}\mathbb{Z} \times \left(\mathbb{Z}/\mathrm{n}\mathbb{Z}\right)^* \to \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ dada por $(x, r) \mapsto (1 + n)^x r^{n^s} \bmod n^{s+1}$ es un isomorfismo, y además se cumple que $\psi_s(x_1 + x_2 \bmod n^s, r_1 r_2 \bmod n) = \psi_s(x_1, r_1)\psi_s(x_2, r_2) \bmod n^{s+1}$.*

Demostración: Para hacer esta demostración empezaremos por definir una aplicación $\pi : (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* \to \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ dada por $r \mapsto \psi_s(0,r) = r^{n^s} \bmod n^{s+1}$. Por como hemos enumerado anteriormente las clases laterales es suficiente con probar que $\pi(r_1 r_2 \bmod n) = \pi(r_1)\pi(r_2) \bmod n^{s+1}$ y que π es inyectiva de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ a H. Con estas dos condiciones tenemos: por un lado la prueba de que ψ_s es biyectiva, ya que es una aplicación entre espacios con el mismo cardinal y si podemos llevar $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ de forma inyectiva a H mediante π, significa que cada una de las clases laterales anteriores (que son disjuntas) se puede tener cambiando el primer argumento de ψ_s por como está definida, luego sería biyectiva. Además, el hecho de que ψ_s es homomorfismo de grupos resulta de $(1+n)^{x_1}(1+n)^{x_2} = (1+n)^{x_1+x_2} = (1+n)^{\alpha+n^s} = (1+n)^{\alpha} = (1+n)^{x_1+x_2 \bmod n^s}$ y la segunda componente se prueba directamente de probarlo para la aplicación π. Lo primero en lo que nos tenemos que fijar es que $\pi(r) \in H$ de forma obvia porque esa imagen forma parte de la clase lateral $(1+n)^0 H$ y por tanto todas sus potencias también. Para probar la propiedad homomórfica vamos a usar la expresión del binomio de Newton de la siguiente forma, en primer lugar vamos a poner el representante en $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ de la multiplicación de dos elementos $r_1 r_2 \equiv \beta \bmod n$ luego $r_1 r_2 = \beta + k \cdot n$, porque si multiplicamos los argumentos antes de introducirlos en la función como están en $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ nos van a quedar reducidos módulo n. Tendríamos que $\pi(r_1)\pi(r_2) \equiv r_1^{n^s} r_2^{n^s} \equiv (r_1 r_2)^{n^s} \equiv (\beta + k \cdot n)^{n^s} \equiv \beta^{n^s} \equiv (r_1 r_2 \bmod n)^{n^s} \equiv \pi(r_1 r_2 \bmod n) \bmod n^{s+1}$. Entonces como dos elementos solo son iguales por la aplicación si son iguales módulo n, tenemos que es inyectiva. Sea $\pi(r_1) \equiv \pi(r_2) \bmod n^s$ entonces $r_1 \equiv r_2 \bmod n$ por lo que la aplicación es claramente inyectiva y por el principio del palomar tenemos que es biyectiva. □

Es obvio de lo que acabamos de probar que: $\overline{(1+n)^i} = \psi_s(i, (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*)$. Lo único que necesitamos ahora es calcular el inverso de ψ_s. En este caso no va a ser tan sencillo como en el cifrado de Paillier original, sin embargo sí que se puede invertir con un proceso rápido y algorítmico sabiendo la factorización de n.

Teorema 4.1. *Para cualquier n admisible y $s < p, q$, la aplicación $\psi_s : \mathbb{Z}/\mathrm{n}^{\mathrm{s}}\mathbb{Z} \times (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* \to \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ dada por $(x,r) \mapsto (1+n)^x r^{n^s} \bmod n^{s+1}$ puede ser invertida en tiempo polinómico dado $\lambda(n)$.*

Demostración: Primero vamos a mostrar como encontrar i a partir de $(1+n)^i \bmod n^{s+1}$. Si definimos la función $L : S_n \to \mathbb{Z}/\mathrm{n}^{\mathrm{s}}\mathbb{Z}$ de la misma forma que la definió Paillier en su cifrado original, solo que trasladándola al espacio correspondiente, tenemos $L(b) = \frac{b-1}{n}$ y entonces es claro que:

$$L((1+n)^i \bmod n^{s+1}) = \left(i + \binom{i}{2} n + \cdots + \binom{i}{s} n^{s-1}\right) \bmod n^s$$

La idea del algoritmo que queremos es sacar el valor parte a parte, de forma que primero se extrae $i_1 = i \bmod n$, luego $i_2 = i \bmod n^2$ y así sucesivamente. Es muy fácil conseguir el primer elemento $i_1 = L((1+n)^i \bmod n^2) = i \bmod n$. Ahora podemos conseguir el resto mediante el siguiente paso de inducción: supongamos que conocemos hasta i_{j-1} entonces podemos proceder con el paso j-ésimo, esto significa que $i_j = i_{j-1} + k \cdot n^{j-1}$

para algún $0 \leq k < n$. Si usamos esto en:

$$L((1+n)^i \bmod n^{j+1}) = \left(i_j + \binom{i_j}{2} n + \cdots + \binom{i_j}{j} n^{j-1}\right) \bmod n^j$$

Ahora vamos a fijarnos en la forma en la que está definido i_j para observar que $\binom{i_j}{t+1} n^t$ para $j > t > 0$ cumple que $\binom{i_j}{t+1} n^t = \binom{i_{j-1}}{t+1} n^t \bmod n^j$. Esto es claro porque las contribuciones de la parte $k \cdot n^{j-1}$ se anulan módulo n^j cuando multiplicamos por n, por eso t tiene que ser mayor estricto que 0.

$$\begin{aligned} \binom{i_j}{t+1} n^t &= \frac{(i_{j-1}+kn^{j-1})!}{(i_{j-1}+kn^{j-1}-(t+1))!(t+1)!} n^t = \\ &= \frac{(i_{j-1}+kn^{j-1})(i_{j-1}+kn^{j-1}-1)\ldots(i_{j-1}+kn^{j-1}-(t+2))}{(t+1)!} n^t \end{aligned}$$

Todos los sumandos fruto de productos cruzados del término kn^{j-1} al multiplicar por n^t se van a anular módulo n^j, por tanto se pueden eliminar y es claro que queda $\frac{(i_{j-1})(i_{j-1}-1)\ldots(i_{j-1}-(t+2))}{(t+1)!} n^t = \binom{i_{j-1}}{t+1} n^t$ Esto significa que obtenemos:

$$L((1+n)^i \bmod n^{j+1}) = \left(i_{j-1} + k \cdot n^{j-1} + \binom{i_{j-1}}{2} n + \cdots + \binom{i_{j-1}}{j} n^{j-1}\right) \bmod n^j$$

Entonces simplemente tenemos que:

$$\begin{aligned} i_j &= i_{j-1} + k \cdot n^{j-1} \\ &= i_{j-1} + L((1+n)^i \bmod n^{j+1}) - \left(i_{j-1} + \binom{i_{j-1}}{2} n + \cdots + \binom{i_{j-1}}{j} n^{j-1}\right) \bmod n^j \\ &= L((1+n)^i \bmod n^{j+1}) - \left(\binom{i_{j-1}}{2} n + \cdots + \binom{i_{j-1}}{j} n^{j-1}\right) \bmod n^j \end{aligned}$$

Ahora que ya sabemos recuperar el valor del exponente de $(1+n)^i$ lo que tenemos que hacer es recuperarlo de la imagen de la función. Supongamos que tenemos $c = (1+n)^i r^{n^s} \bmod n^{s+1}$. Vamos a ver cómo conseguir invertir la función por completo, es decir obtener i y r, usando λ. Lo primero que tenemos que hacer es elevar c a λ de forma que obtenemos lo siguiente:

$$c^\lambda = (1+n)^{i\lambda \bmod n^s} r^{n^s \lambda \bmod n^s \lambda} = (1+n)^{i\lambda \bmod n^s}$$

Como $orden((n+1)) = n^s$ si $i\lambda \equiv \alpha \bmod n^s$ entonces $(1+n)^{i\lambda} = (1+n)^{\alpha + kn^s} = (1+n)^\alpha (1+n)^{kn^s} = (1+n)^\alpha$, por eso el módulo en el exponente. Lo mismo sucede con r salvo que en este caso es con $n^s\lambda$ por el teorema de Carmichael 2.4. De lo expuesto anteriormente es posible obtener $i\lambda \bmod n^s$ y de ahí podemos extraer i fácilmente. Para obtener el valor de r una vez conocido i, lo primero que tenemos que hacer es eliminar la otra parte multiplicando por el inverso $r^{n^s} = c(1+n)^{-i} \bmod n^{s+1}$. Una vez tenemos esto así necesitaremos encontrar un elemento $a \in \mathbb{Z}$ tal que se cumple $a\lambda + 1 \equiv 0 \bmod n^s$, el cual existe porque $\text{mcd}(\lambda, n^s) = 1$ y se tiene por la identidad de Bezout. Entonces

recuperar r será simplemente:

$$\left(r^{n^s}\right)^{\frac{a\lambda+1}{n^s}} \bmod n = r^{a\lambda+1} \bmod n = (r^\lambda)^a r \bmod n = r \bmod n = r \qquad \square$$

Lo visto arriba se puede resumir en el siguiente algoritmo para obtener i:

Algorithm 4 Algoritmo para obtener i

1: $recuperaexponentes(a)$:
2: $i \leftarrow 0$ ▷ Inicialización de la variable.
3: **for** $j \leftarrow 1$ **to** s **do**
4: $t_1 \leftarrow L(a \bmod n^{j+1})$ ▷ Valor de la función L.
5: $t_2 \leftarrow i$ ▷ El contador que va a ir bajando para calcular el factorial.
6: **for** $k \leftarrow 2$ **to** j **do**
7: $i \leftarrow i - 1$
8: $t_2 \leftarrow t_2 \cdot i \bmod n^j$ ▷ Hace falta un elemento más del factorial cada vez.
9: $t_1 \leftarrow t_1 - \frac{t_2 \cdot n^{k-1}}{k!} \bmod n^j$ ▷ Quitamos uno a uno cada uno de los elementos de la suma anterior, de la potencia más alta a la menos alta.
10: **end for**
11: $i \leftarrow t_1$ ▷ Cuando el For se haya recorrido todas la veces este i será el que estábamos buscando.
12: **end for**

Con este teorema ya podemos definir un sistema criptográfico completo. Este dependerá del s elegido que habrá sido fijado anteriormente y ya será sabido por ambas partes. Podemos llamar a estos sistemas DJN_s.

Lo primero que vamos a hacer es describir la generación de claves. Necesitamos un parámetro de seguridad k y un número $n = p \cdot q$ admisible, de forma que n tenga k bits. A continuación, tenemos que elegir el parámetro $g \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ que será parte de la clave pública de nuestro cifrado. Este número lo vamos a elegir como $g = (1+n)^j x$ donde j es un entero primo con n y $x \in H$, es decir es un elemento del grupo $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$. Ahora para la parte de la clave privada necesitamos calcular λ. Con este λ tenemos que tomar un elemento d que cumpla las siguiente condiciones. $d \bmod n \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y $d \equiv 0 \bmod \lambda$. Una elección posible es tomar $d = \lambda$ como en el cifrado de Paillier original, sin embargo, cualquier d que cumpla lo anterior es válida. En resumen tenemos nuestra clave pública que sería: n y g; y nuestra clave privada que sería: d.

La siguiente parte del sistema criptográfico es el algoritmo de cifrado. Supongamos que tenemos un mensaje $m \in \mathbb{Z}/\mathrm{n}^{\mathrm{s}}\mathbb{Z}$; el cual va a ser cifrado usando la clave pública del receptor. Elegimos un elemento aleatorio de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ que vamos a denotar por r y calculamos el mensaje cifrado como $c = \mathcal{E}_{DJN_s}(m; n, g) = g^m r^{n^s} \bmod n^{s+1}$.

Lo único que nos queda es el algoritmo de descifrado. Partimos de un mensaje $c \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ que acabamos de recibir de un emisor. Lo primero que hacemos es calcular $c^d \bmod n^{s+1}$ porque esto tiene el siguiente efecto en el cifrado: $c^d \equiv (1+n)^{m\,d\,j} x^{m\,d} r^{n^s d} \equiv$

$(1+n)^{m\,d\,j}(x^m r^{n^s})^d \equiv (1+n)^{m\,d\,j} = (1+n)^{m\,d\,j \bmod n^s}$. Usando el algoritmo 4, es posible entonces obtener el valor de $m\,d\,j \bmod n^s$. Por otro lado, si hacemos lo mismo con la g podemos recuperar $d\,j$, si hemos elegido nosotros tanto la d como la j no nos hace falta calcularlo. Además siempre sería la misma cuenta así que se puede guardar directamente como parte de la clave privada. Una vez tenemos ambas cosas simplemente calculamos $(m\,d\,j)(d\,j)^{-1} \equiv m \bmod n^s$.

Este sería la modificación de Damgård-Jurik-Nielsen al cifrado de Paillier completa. Al igual que en Paillier se suele coger $g = (1+n)$ de forma habitual.

4.1.2. Seguridad del nuevo esquema

Tenemos que saber, para verificar si el sistema es seguro, si se mantiene que la función es de una vía. Además, igual que le pasaba al cifrado original de Paillier, es aleatoriamente auto-reducible, se puede ver exactamente cómo en [4]. Entonces, como un problema se puede reducir en tiempo polinómico a otro aleatorio, si estas propiedades fallan en una cantidad no despreciable de problemas, no se puede considerar que el sistema es seguro. En caso de encontrarnos con un problema que no es de ese subconjunto se podría reducir a otro, tantas veces como fuera necesario, que sí lo sea. Lo primero de todo en lo que nos tenemos que fijar es en la siguiente proposición:

Proposición 4.1. *Si para algún $t \in \mathbb{N}$ se cumple que DJN_t es tiene una función trampa, entonces para todo $s > t$ también se cumple que DJN_s es de una vía.*

Demostración: Partamos de que el cifrado para un $t \in \mathbb{N}$ es de una vía, entonces supongamos que existe un $s > t$, para el cual el cifrado no es de una vía. Entonces supongamos que tenemos un mensaje $m \in (\mathbb{Z}/n^t\,\mathbb{Z})$, el cual ha sido cifrado usando la clave pública n, $g \in (\mathbb{Z}/n^{t+1}\,\mathbb{Z})^*$ y se obtiene $c_t \in (\mathbb{Z}/n^{t+1}\,\mathbb{Z})^*$. Vamos a pasar el mensaje c_t a módulo n^{s+1} y descifrarlo como si fuera un mensaje de $\mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ que ha sido cifrado usando n y una $g' \in \left(\mathbb{Z}/\mathrm{n^{s+1}}\mathbb{Z}\right)^*$ que cumpla que $g' \equiv g \bmod n^{t+1}$ como claves públicas. Como hemos supuesto que el problema sobre $\mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ no es de una vía, simplemente tendremos descifrar c_t o reducirlo a otras instancias aleatorias hasta que una se pueda descifrar y así obtener el mensaje $m \in \mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ que corresponde a $c_t \in \left(\mathbb{Z}/\mathrm{n^{s+1}}\mathbb{Z}\right)^*$. Sin embargo es claro que $m \bmod n^t$ es el mensaje que se había cifrado en c_t originalmente, por lo tanto DJN_t no es de una vía y se tiene una contradicción. □

Este resultado es especialmente interesante porque el cifrado original de Paillier es DJN_1, por tanto, si el sistema original es de una vía significa que estos criptosistemas son de una vía para todas las $s \in \mathbb{N}$.

En conclusión, en esta sección hemos visto una modificación al criptosistema de Paillier propuesta por Damgård, Jurik y Nielsen que permite cifrar mensajes mucho más largos con la misma n y que reduce el factor de expansión de 2 a 1 y poco, depende del s. Este factor de expansión hace referencia a la cantidad de bits que ocupa el mensaje. En Paillier original como el mensaje cifrado ocupa n^2 frente a n que ocupa el mensaje en claro, su factor de expansión es $\frac{log_2(n^2)}{log_2(n)} = 2$ y pasa a ser $\frac{s+1}{s}$.

4.2. Otras modificaciones

Además de esta modificación que acabamos de ver existen otras que también se han desarrollado a partir del criptosistema que publicó Paillier en 1999. En [2] podemos ver un resumen de varias de estas variantes. No son tan útiles como el esquema de cifrado original porque este tiene alguna característica extra que las modificaciones no, sin embargo no se van a tratar en el trabajo. Aún así vamos a hablar de dos de ellas para hacernos una idea de como se puede modificar este cifrado y de lo flexible que puede llegar a ser. La primera que vamos a comentar está creada por el mismo Paillier y se habla de ella en el mismo artículo donde propone el cifrado [11], la segunda de la que vamos a hablar fue creada por Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham y Phong Q. Nguyen [3] en 2001. Hemos elegido estas dos porque la primera ha sido propuesta por el propio creador del criptosistema original y la segunda porque es la que más parece que se aleja del cifrado original.

4.2.1. Variante 1: Paillier

Esta variante nace del intento de reducir la complejidad del algoritmo de descifrado. La complejidad del algoritmo de descifrado original es $O(n^3)$ y con esta modificación se puede llegar a tener una complejidad de $O(n^{2+\epsilon})$. Este interés en reducir el tiempo de descifrado, incluso sacrificando parte de la seguridad, es debido a que en 1999 ningún criptosistema tenía una velocidad de descifrado tan alta y era un punto muy fuerte a favor. Sin embargo, ahora el cifrado se puede romper resolviendo un problema más sencillo que el CR[n] que hacía falta en Paillier original.

En este cifrado modificado necesitaremos en primer lugar un elemento $g \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ porque restringe el número de posibles mensajes cifrados de $\left|\left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*\right|$ a solamente el número de elementos en el subgrupo generado por g, $|\langle g\rangle|$. Al igual que en el cifrado original, es necesario que este elemento g tenga orden múltiplo de n, es decir αn para algún $1 \leq \alpha \leq \lambda$. Entonces el esquema de cifrado es el siguiente, lo primero que necesitamos es un n admisible y un elemento $g \in \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ con $ord_{n^2}(g) = \alpha n$. Entonces ya tenemos las claves: la clave pública sería n y g y la clave privada sería α. El esquema de cifrado es más sencillo que el de Paillier original, supongamos que tenemos un mensaje $m < n$, tomamos un número aleatorio $r < n$ y entonces ciframos como: $\mathcal{E}(m; n, g) = g^{m+nr} \bmod n^2 = c$. El esquema de descifrado es el siguiente, $m = \frac{L(c^\alpha \bmod n^2)}{L(g^\alpha \bmod n^2)} \bmod n$. Entonces tenemos que se cumple que para todo $\omega \in \langle g\rangle$

$$[\![\omega]\!]_g = \frac{L(\omega^\alpha \bmod n^2)}{L(g^\alpha \bmod n^2)} \bmod n$$

Ahora solo nos queda ver por qué funciona.

En primer lugar queremos ver que $g^\alpha \equiv 1 \bmod n$, supongamos por reducción al absurdo que $g^\alpha \equiv t \bmod n$ con $t \neq 1$. Entonces $g^\alpha \equiv t + nk \bmod n^2$ y sucedería lo siguiente:

$$1 \equiv g^{\alpha n} \equiv (t+nk)^n \equiv t^n + t^{n-1}nk \cdot n + \cdots \equiv t^n \bmod n^2 \text{ Contradicción}$$

1 es un residuo n-ésimo módulo n^2, y como tal existe un único $0 < a < n$ tal que $a^n \equiv 1 \mod n^2$. Por tanto es claramente 1 y esto es una contradicción con la suposición. Por tanto $g^\alpha \equiv 1+kn \mod n^2$. Sabiendo esto vamos a desarrollar lo que sucede al aplicar la función $L(\cdot)$ a ambos elementos ω^α y g^α:

$$\omega^\alpha \equiv (g^{m+nr})^\alpha \equiv g^{m\alpha+rn\alpha} \equiv g^{m\alpha}g^{rn\alpha} \equiv (1+nk)^m \equiv \boxed{\text{por 3.1}} \equiv 1+mnk \mod n^2$$

$$g^\alpha \equiv 1+nk \mod n^2$$

Por tanto, nos queda que $\frac{L(\omega^\alpha \mod n^2)}{L(g^\alpha \mod n^2)} = \frac{mk}{k} = m \mod n$, luego efectivamente el descifrado funciona bien.

La operación más costosa de todo el cálculo es c^α que es la que nos va a dar la complejidad. Esta operación tiene una complejidad de $O(|n|^2|\alpha|)$, si se elige α de tal forma que $|\alpha| = |n|^\epsilon$ se tiene la complejidad $O(|n|^{2+\epsilon})$.

En cuanto a la seguridad este cifrado se puede romper si se resuelve el problema parcial del logaritmo discreto (*Partial Discrete Logarithm Problem*, PDL).
Definición 4.1. El problema PDL[n,g] se define de la siguiente manera: dado $\omega \in \langle g \rangle$, calcular $[\![\omega]\!]_g$.

Es obvio que se rompería el cifrado si pudiésemos encontrar el valor de $N \in \mathbb{Z}$ tal que $g^N = \omega \mod n^2$. A partir de él se puede escribir $N = m + nr$ y ya tendríamos el mensaje.
Definición 4.2. El problema de decisión asociado a PDL[n,g] será denotado como D-PDL[n,g] y se define como: dado $\omega \in \langle g \rangle$ y $x \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$, determinar si $[\![\omega]\!]_g = x$.

Si alguno de los dos fallase se tendría que el cifrado se puede romper, entonces este esquema es seguro si y solamente si PDL[n,g] es un problema computacionalmente difícil.

4.2.2. Variante 2: Catalano-Gennaro-Howgrave-Nguyen

Dos años después del artículo de Paillier aparece el artículo de los autores del cifrado del que vamos a hablar en este apartado. En su *paper* se propone un esquema basado en el de Paillier pero permitiendo un exponente arbitrario más pequeño en lugar de n, esto es muy interesante por como afecta a las condiciones de seguridad del cifrado y otros aspectos del mismo. En particular permite agilizar en gran medida el tiempo de cifrado y deja prácticamente igual el de descifrado. El cambio en el nivel de seguridad es obvio porque ahora el problema que permitiría romper el cifrado es decidir si un elemento es un residuo e-ésimo, siendo e la potencia más pequeña que vamos a tomar aquí para cifrar. Ahora la seguridad del cifrado descansa completamente en la incapacidad por parte de un usuario de determinar la raíz e-ésima de un elemento en $\mathbb{Z}/\mathrm{n}^2\mathbb{Z}$. Lo que habíamos denotado por RSA[n,e].

Para esta modificación del cifrado necesitaremos en primer lugar un n admisible como siempre. A continuación vamos a necesitar un número $e \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$ tal que mcd$(e, \lambda(n^2)) = 1$, y por tanto existe su inverso $d \in \mathbb{Z}$ cumpliendo $d \cdot e \equiv 1 \mod \lambda(n^2)$. Con esto tenemos

todo lo que necesitamos para el cifrado y las claves tanto públicas como privadas: la clave pública será n y e; y la clave privada será d.

Ahora vamos a presentar la función de cifrado para un mensaje $m \in \mathbb{Z}/\mathrm{n}\mathbb{Z}$, $\mathcal{E}(m; n, e) = (1 + n)^m r^e \bmod n^2$. Esta función es obviamente biyectiva en $\mathbb{Z}/\mathrm{n}\mathbb{Z} \times (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* \longrightarrow \left(\mathbb{Z}/\mathrm{n}^2\mathbb{Z}\right)^*$ con e fijo. La prueba es prácticamente igual que la que se hizo con el sistema de Paillier original.

Ahora veamos el sistema de descifrado. Partamos de un texto cifrado $c = (1 + n)^m r^e$ y empecemos a descifrarlo, lo primero que vamos a hacer es calcular:

$$(c \bmod n)^d \bmod n \equiv ((1 + n)^m r^e)^d \equiv (1 + nmd) r^{ed} \equiv r^{ed} \equiv r^{1-\lambda(n^2)k} \equiv r \bmod n \quad (4.1)$$

Una vez obtenido r lo único que tenemos que hacer para descifrar el mensaje por completo es calcular $c/r^e \equiv (1 + mn) \bmod n^2$, y a partir de aquí determinar n. Juntando todo la función de descifrado queda $\mathcal{D}(c; n, d) = \frac{\frac{c}{(c^d \bmod n)^e} - 1 \bmod n^2}{n}$.

Hay que notar que si hacemos c módulo n nos va a quedar $r^e \bmod n$, luego si podemos resolver RSA[n,e] no es necesario tener d para descifrar c. La función de cifrado dejaría de ser una función trampa y ya no funcionaría el criptosistema.

Capítulo 5

Voto electrónico

La democracia y la toma de decisiones mediante la votación es una dinámica muy antigua y extendida. En cierto modo es la forma más justa de tomar una decisión teniendo en cuenta la opinión de todas las partes involucradas. Como era de esperar en la época moderna se siguen realizando votaciones, solo que ahora se realizan también de forma online. El voto electrónico surge a raíz de la invención de Internet y tiene como principal ventaja que puede realizarse sin que las partes se desplacen a un lugar y gastando menos tiempo y dinero en infraestructuras y material. La principal desventaja es la falta de confiabilidad que se pueda tener en un voto emitido de forma remota. Aquí es donde entra la criptografía y el desarrollo de las firmas digitales y los protocolos de verificación. En concreto nosotros hablaremos de cómo usar el cifrado de Paillier para realizar una votación y, a su vez, asegurar la validez de la misma. En esta sección hemos consultado sobre todo [12], [9] y [6], en ellos se explica de forma muy similar las partes de una votación así como los protocolos de seguridad.

5.1. Elementos de una votación

En general se va a considerar que un proceso de votación consta de tres partes, la fase de preparación, la fase de votación y la fase de recuento.

- **Fase de preparación**: En esta fase habría que registrar los votantes, reservar el sitio donde se va a realizar la votación, imprimir los votos, conseguir una urna o recipiente para guardarlos... En el voto electrónico esta fase simplemente consiste en la identificación de los votantes y en la generación de las claves del cifrado que se vayan a usar en la votación.
- **Fase de votación**: En esta fase cada votante se identificaría en el correspondiente lugar de voto e introduciría el suyo en la urna siendo todo esto supervisado. En el caso electrónico la persona elegirá al candidato o candidatos que desee votar y cifrará su voto antes de subirlo a un espacio público en el que se van a ir colocando los votos, generalmente con una prueba de corrección. Vamos a asumir que el número de veces que vota una persona está de alguna forma controlado por el

espacio al que se suben los votos.

- **Fase de recuento**: Esta fase se puede hacer de varias formas dependiendo de la información adicional al resultado que queramos. Se puede hacer un recuento de cada urna y sumar los votos, de esta forma se pueden tener datos de como se distribuyó la tendencia electoral por los diferentes puntos del territorio. O se juntan los votos de todas las urnas y hay un grupo de personas encargadas de contarlos todos juntos, esto es en general más económico y es más fácil prevenir comportamientos fraudulentos al estar todo centralizado. Para el voto electrónico se hace uso de que el cifrado es homomórfico y se operan todos los mensajes cifrados con los votos para hacer el recuento. Una vez obtenido el valor del recuento cifrado las autoridades encargadas de la votación procederán al descifrado del mismo y la publicarán en el espacio público donde se han ido subiendo los votos, esto bien puede ser una página web, un servidor... Nosotros lo llamaremos espacio público en general.

Aquí es donde se empieza a ver claro porque el cifrado de Paillier cumple muy bien las exigencias que se le piden para una votación electrónica. En primer lugar tiene la propiedad homomórfica necesaria para hacer el recuento sin tener que descifrar los votos individualmente, lo cual garantiza privacidad porque si no las autoridades sabrían cada voto y si este está ligado a la persona de alguna forma se sabría que votó cada uno. Pero también tiene una propiedad muy importante, el cifrado es aleatorio, el elemento r del cifrado es un elemento aleatorio de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$, por tanto, cada vez que cifras un mensaje el texto cifrado es diferente aunque estés cifrando el mismo mensaje. Esto es primordial para garantizar la privacidad del voto, si no en cuanto se supiera el resultado de la votación se podría asociar cada texto cifrado con el mensaje que cifra y se podría, de nuevo, identificar que votó una persona si se puede averiguar cual es su voto. Además, la propiedad homomórfica no es necesaria solo para el recuento, sino también para los protocolos que veremos.

Entroncando con esto vamos a hablar de qué requisitos tiene que cumplir una votación empezando por el ya comentado, la privacidad. Durante el proceso ningún participante en él puede obtener información de un voto particular de otra persona. Solo se conocerá el resultado general de la votación y en algunos casos los votos cifrados de los participantes, pero en ningún caso esto puede suponer que luego se pueda relacionar con el voto de alguien. Otro aspecto clave del proceso es la completitud, el número de votos contados tiene que ser consistente con el número de votos emitidos. Además es importante la robustez del proceso, poder garantizar que el resultado solo está conformado de los votos que han sido emitidos de forma correcta, incluso si algunos votantes o autoridades intentan engañar en el proceso. Por último, verificabilidad universal, el resultado tiene que poder ser verificado por cualquier persona.

Ahora solamente nos faltaría hablar de las partes implicadas en el proceso:

- **Votantes**: El límite superior de votantes va a estar denotado por W, un votante no tendría porque estar obligado a participar. A cada uno de los votantes lo de-

notaremos por W_i y al total de votos emitidos como v. También suponemos que cada votante tiene una capacidad de almacenamiento de información que solo es accesible a él.

- **Autoridades**: Las autoridades totales se denotarán por A, cada una como A_i y al número de autoridades honestas lo vamos a denotar por H. Estas autoridades honestas suponemos que hacen su trabajo de forma correcta.
- **Opciones de voto**: A las opciones de voto las vamos a llamar candidatos y denotaremos por L al número total de candidatos y a cada uno de ellos como L_i. Si $L = 2$ llamaremos a esa votación una votación sí/no y generalmente se tomará "1" sí y "0" no.

En el trabajo vamos a cubrir tres tipos de votaciones: en primer lugar las de sí/no; luego las que requieren elegir un candidato de entre L posibles; y las que hay que elegir a lo sumo k candidatos de los L posibles. Además también hay varios protocolos dependiendo de cómo de deshonestas sean las autoridades y los votantes. El menor problema viene cuando tanto las autoridades como los votantes son honestos, pero curiosos. Esto va escalando hasta protocolos que garantizan que la votación también puede tener lugar aunque se emitan votos corruptos y un cierto número de autoridades también sean corruptas. En general el esquema de Shamir no es muy bueno porque cualquier subconjunto de un conjunto autorizado obtiene el secreto. Sin embargo, como aquí lo vamos a usar para un descifrado umbral ninguna de las autoridades va a conocer el secreto, simplemente van a poder usarlo al combinarse, de forma que aunque participe una autoridad corrupta no obtiene nada de información del secreto.

5.2. Protocolos de voto con votantes y autoridades honestas

La situación ideal de la votación sería que tanto los votantes como las autoridades fueran honestos y no curiosos, es decir, hicieran bien las cosas y no intentaran consultar el voto de los demás. Nuestra votación siempre se tiene que aproximar lo más posible a una votación ideal, aquí es donde entran en juego el cifrado y los protocolos. En esta primera situación no vamos a necesitar protocolos para que los votantes demuestren que realmente han elegido uno de las posibles candidatos, simplemente con el cifrado es suficiente.

Votación sí/no

Usaremos el esquema de cifrado de Damgård-Jurik-Nielsen en 4.1, por lo tanto lo primero que necesitaremos es un $n \in \mathbb{N}$ admisible y un s, además de la clave privada d que recordemos tiene que cumplir que $d \bmod n \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y $d \equiv 0 \bmod \lambda$. La clave privada se proporcionará a las autoridades mediante reparto de secretos como se explica en 2.2.1 y la clave pública la tendrán tanto los votantes como las autoridades. En esta votación $L = 2$ y cada votante W_i tiene que elegir entre $m_i = 0$ para votar no y $m_i = 1$ para votar sí.

Una vez terminada la fase de preparación empieza la fase de votación. En esta fase cada votante va a emitir su voto el cual va a cifrar siguiendo DJN_s. Es decir, el participante W_i va a elegir su voto m_i, va a calcular $\mathcal{E}_{DJN_s}(m_i; n) = c_i$ y va a guardar en su directorio privado el elemento aleatorio $r_i \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ que usó para cifrar el voto. A continuación, va a publicar c_i en el espacio público habilitado para realizar la votación.

Cuando termina la fase de votación y todos los participantes ya han votado, se comienza con la fase de recuento. Las autoridades encargadas de realizar el recuento van a coger del espacio público todos los votos cifrados y van a realizar la siguiente operación: $\prod_{i=1}^{W} c_i = c$. Este mensaje cifrado c es el que va a contener el recuento total de los votos y es el que hay que descifrar. Recordemos cómo era la propiedad homomórfica del cifrado de Paillier, la multiplicación de los mensajes cifrados al descifrar nos da la suma de los mensajes en claro. Entonces lo siguiente que hacen las autoridades es $\mathcal{D}_{DJN_s}(c; n, d) = m$ y m es el resultado de la votación que será publicado en el mismo sitio donde estaban subidos los votos junto con c. En concreto como los síes se ponían como 1, $m =$ votos positivos y $W - m =$ votos negativos.

Como podemos ver en este caso con votantes honestos, podemos estar seguro de que el sistema es completo y robusto. Como los votantes son honestos podemos estar seguros de que ningún votante votó más de una vez y además, como son honestos, no hay necesidad de verificar si los votos son válidos. Además es privado porque todos los votos se suben cifrados y las únicas que pueden descifrarlos son las autoridades de la votación, pero una sola de forma individual no puede descifrar un voto. De esta forma está garantizada la privacidad a pesar de que los votantes y las autoridades sean curiosas. Además, se publica c junto con m, para que el resultado de la votación pueda ser verificado por cualquiera que quisiera revisarlo.

Votación eligiendo 1 de entre L candidatos

No nos vamos a detener en detalles de los aspectos que son similares al anterior. Vamos a centrar la explicación en qué es diferente del anterior. En primer lugar, la fase de preparación es muy similar, necesitamos generar las claves del cifrado DJN_s. Sin embargo, como ahora hay L candidatos los votos no van a ser tan simples como 0 y 1. Un voto por el candidato j se va denotar V^j y hay $\{0, 1, \ldots, L-1\}$ posibles votos. Entonces, ahora el voto del votante i será $m_i = V^j$ para algún $0 \leq j < L$. Aquí ya hay que tener un poco más de cuidado eligiendo s que antes ya que hay que tener en cuenta que, si V^{L-1} es el mayor de los votos, entonces $n^s < W \cdot V^{L-1} < V^L$. Luego en la fase de recuento hablaremos más sobre que número se debería coger como V. Entonces $s < L \cdot \log_n(W)$.

La fase de votación es exactamente igual. El votante i emitirá su voto $m_i = V^j$ y procederá a cifrarlo mediante la función $\mathcal{E}_{DJN_s}(V^j; n) = c_i$. Se guarda para sí la $r_i \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ con la que cifró y publica en el espacio habilitado c_i.

La fase de recuento se realiza de la misma forma que antes, las autoridades van a multiplicar todos los votos cifrados para obtener el resultado cifrado c y van a calcular $\mathcal{D}_{DJN_s}(c; d) = m$. Pero, ¿qué es exactamente m en este caso? En este caso

$m = \sum_{j=0}^{L-1} v_j V^j$, donde v_j representa el número total de votos que ha recibido el candidato $j-$ésimo. Nosotros obtenemos la suma solamente, para poder recuperar el número de votos de cada candidato hay que elegir de forma adecuada los V^j para poder resolver el problema de la mochila 2.3. Si el número total de votantes es W, usaremos como votos las potencias de un número $a > W$. De esta forma un voto por el candidato $j-$ésimo sería $m_i = a^j$. Es muy importante que a sea mayor que W porque así $W \cdot a^j < a^{j+1}$, y la suma resultante de los votos, $v_0 + v_1 \cdot a + \cdots + v_{L-1} \cdot a^{L-1}$, tendrá el mismo resultado que el problema de al mochila usando como pesos los a^j. Observemos que en caso contrario si $W \cdot a^j > a^{j+1}$, al resolver, la solución óptima será algo tipo $v_j < W$ y $v_{j+1} = 1$; lo cual ya se ve que no funciona porque falla en la completitud. En [6] Jurik toma los votos como potencias de 2 y luego eleva cada voto a una potencia j tal que $2^j > W$, para evitar que los votos se pasen de un candidato a otro como acabamos de ilustrar. Esta solución es menos intuitiva, así que nos quedamos con la anteriormente explicada. En este caso las autoridades publicarán en el espacio público con los votos tanto c como m y los votos de cada candidato, es decir el resultado de la votación.

Ejemplo 5.1. Supongamos que tenemos una votación con 4 candidatos, $\{0, 1, 2, 3\}$, y 7 votantes. En primer lugar hay que elegir un a adecuado para codificar el voto, por ejemplo $a = 8 > 7$. Entonces tenemos el siguiente resultado de la votación después de descifrar el resultado $m = 1162 < 8^4 = 4096$. Si resolvemos el problema de la mochila con elementos de peso $8^i,\ i \in \{0, 1, 2, 3\}$, nos queda $2 \cdot 8^3 + 2 \cdot 8^2 + 1 \cdot 8^1 + 2 \cdot 8^0 = 1162$. Luego el resultado de la votación es 2 votos para los candidatos 0, 2 y 3 y un voto para el candidato 1.

Votación eligiendo a lo sumo k de entre L candidatos

Para este último caso nos encontramos en una situación muy parecida a la anterior. Lo primero que tenemos que hacer de igual forma que antes es generar las claves del criptosistema. La única diferencia con el anterior es que ahora cada votante i puede realizar hasta k votos, no necesariamente k. Para englobar todos los posibles casos se van a añadir $k - 1$ candidatos ficticios, de forma que si un votante sólo escoge i candidatos, los $k-i$ votos sobrantes que le corresponderían se van a añadir a algunos de los $k - 1$ candidatos ficticios. Ahora el número de candidatos es un conjunto más amplio $\{0, 1, 2, \ldots, L + k - 2\}$, de forma que existen más valores para V^j y esto repercutirá a la forma de elegir s. En esta votación se tiene que cumplir, de la misma forma que el caso anterior, que $V^{L+k-2} < n^s$; lo cual implica que $s > (L+k-2)\log_n(V)$, luego es necesario que el valor de s sea más grande.

Una forma de realizar la votación es que cada votante cifre sus votos de forma individual y los suba de uno en uno. Sin embargo, no es la única forma. Una forma más rápida es que cada votante, aprovechando la propiedad homomórfica del cifrado, sume con anterioridad el valor de todos sus votos y cifre directamente el voto total. De esta forma, se reduce el número de operaciones a realizar por parte de las autoridades. Sin embargo, se prefiere la primera opción en lugar de la segunda por varios motivos. El primero de todos es que si los votos se suben de uno en uno es más fácil verificar que se cumple la

completitud y comprobar la robustez es también mucho más fácil. El segundo es que al subir los votos de forma individual y separados también permite que sea mucho más fácil para el propio votante verificar que su voto está bien emitido. La dinámica es la misma, el individuo i con su voto t ha decidido votar a j, entonces se cifrará el mensaje $m_{i_t} = V^j$ como $c_{i_t} = \mathcal{E}_{DJN_s}(V^j; n)$, que es el voto después del cifrado.

Por último la fase de recuento es exactamente igual que la de la votación de 1 entre L. Se multiplican todos los mensajes cifrados para, posteriormente, descifrar el mensaje y obtener el valor de la suma de los votos de todos los votantes. Igual que antes se elegirá de forma adecuada W^j para realizar la misma operación que se explica en el párrafo anterior. Una vez se separan y se sabe cuanta gente votó a cada candidato las autoridades harán públicos estos resultados, junto con c y m.

En todas la votaciones hemos asumido que al menos se vota a uno de los candidatos, si no se quisiera obligar a votar a los participantes se podría añadir un candidato ficticio más simplemente.

5.3. Σ-protocolos

En esta sección vamos a presentar una serie de protocolos. Vamos a explicar como funcionan y donde se añadirían en los esquemas de votación que ya hemos explicado en la sección 5.2. Estas incorporaciones permiten relajar las asunciones de honestidad sobre las partes de la votación, ahora se podrá suponer que hay votantes que engañan y autoridades que también lo hacen.

Tanto los protocolos como las pruebas de conocimiento cero que vamos a presentar aquí tienen un parámetro t, este parámetro representa la longitud del desafío que va a lanzar V. Tiene que cumplir que $2^t < p, q$.

Cifrado del 0

Pongamos en contexto porque es importante el cifrado y el descifrado del 0. Supongamos que tenemos un votante P, el cual quiere asegurar que su mensaje cifrado es uno de los posibles textos cifrados de m en el criptosistema de Paillier, $c = (1+n)^m r^{n^s}$, pero para esto no puede revelar el valor de r. Probar eso es lo mismo que probar que dado $x = c(1+n)^{-m} \bmod n^{s+1}$, este es un cifrado del 0, o que x es una potencia n^s-ésima en $\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}$. El testigo de esta relación será $\omega = r$. Como vamos a necesitar cifrar usando la misma $r \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ que se tome para cifrar el c, introduciremos una función de cifrado exactamente igual que la de Damgård-Jurik-Nielsen solo que con tres argumentos de entrada. $\mathcal{E}_{DJN_s}(m; n, r) : \mathbb{Z}/\mathrm{n}^{\mathrm{s}}\mathbb{Z} \times (\mathbb{Z}/\mathrm{n}\mathbb{Z})^* \longrightarrow \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ con $m \in \mathbb{Z}/\mathrm{n}^{\mathrm{s}}\mathbb{Z}$, $n \in \mathbb{N}$ y $r \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$.

Empecemos describiendo el Σ-protocolo, en este caso el input en común que tendrán el que demuestra y el verificador es $x \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$, n y s. El testigo del secreto que conoce P es $\omega = r \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y lo que se quiere probar es que $x = \mathcal{E}_{DJN_s}(0; n, r)$. La conversación del Σ-protocolo quedará:

1. $P \to V$: P elige un $f \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y manda $a = \mathcal{E}_{DJN_s}(0; n, f)$ a V.
2. $V \to P$: V elige un número aleatorio de t bits e y se lo manda a P.
3. $P \to V$: P manda $z \equiv f\omega^e \mod n$ a V

La verificación por parte de V es la siguiente, primero comprueba que x, a y z son primos con n y acepta la conversación si se cumple que $\mathcal{E}_{DJN_s}(0; n, z) \equiv ax^e \mod n^{s+1}$.

$$\mathcal{E}_{DJN_s}(0; n, z) \equiv (1+n)^0 z^{n^s} \equiv (f\omega^e)^{n^s} \equiv f^{n^s}(\omega^{n^s})^e \equiv ax^e \mod n^{s+1}$$

Observemos que P no revela ω a V en ningún momento, sin embargo si no lo conociera es imposible salvo con una posibilidad despreciable que hubiera podido construir un z verificable.

Proposición 5.1. *El protocolo cumple la solvencia especial.*

Demostración: Supongamos que tenemos dos conversaciones aceptadas (a, e, z) y $(a, \tilde{\mathrm{e}}, \tilde{\mathrm{z}})$ con $\tilde{\mathrm{e}} \neq e$.

Como las conversaciones han sido aceptadas: $\mathcal{E}_{DJN_s}(0; n, z) \equiv ax^e \mod n^{s+1}$ y $\mathcal{E}_{DJN_s}(0; n, \tilde{\mathrm{z}}) \equiv ax^{\tilde{\mathrm{e}}} \mod n^{s+1}$ y usando la propiedad homomórfica del cifrado: $\mathcal{E}_{DJN_s}(0; n, \frac{z}{\tilde{\mathrm{z}}}) \equiv x^{e-\tilde{\mathrm{e}}} \mod n^{s+1}$

Teniendo en cuenta que hemos supuesto que $2^t < p, q$ sabemos seguro que mcd$((e - \tilde{\mathrm{e}}), n) = 1$ y por tanto existen α y β por la identidad de Bezout que cumplen $\alpha n^s + \beta(e - \tilde{\mathrm{e}}) = 1$. Ahora sea $\tilde{x} \equiv x \mod n$ y $\tilde{\omega} = \tilde{x}^\alpha \left(\frac{z}{\tilde{\mathrm{z}}}\right)^\beta \mod n$. Ahora vamos a verificar que este $\tilde{\omega}$ que acabamos de escribir verifica lo mismo que ω.

Notamos que $\mathcal{E}_{DJN_s}(0; n, \tilde{x}) = \mathcal{E}_{DJN_s}(0; n, x \mod n) = x^{n^s}$ y solo nos queda verificar que $\tilde{\omega} \equiv \omega \mod n$:

$$\mathcal{E}_{DJN_s}(0; n, \tilde{\omega}) = \mathcal{E}_{DJN_s}(0; n, \left(\frac{z}{\tilde{\mathrm{z}}}\right)^\beta x^\alpha) = \mathcal{E}_{DJN_s}(0; n, \tilde{x})^\alpha \mathcal{E}_{DJN_s}(0; n, \frac{z}{\tilde{\mathrm{z}}})^\beta = x^{\alpha n^s} x^{\beta(e-\tilde{\mathrm{e}})} = x \mod n^{s+1}$$

Entonces vemos que $x = \mathcal{E}_{DJN_s}(0; n, \tilde{\omega})$ y por tanto $\tilde{\omega} = \omega$. El secreto es el mismo y se puede obtener a partir de los elementos de dos conversaciones (a, e, z) y $(a, \tilde{\mathrm{e}}, \tilde{\mathrm{z}})$ con el mismo a. □

Para terminar de determinar que es Σ-protocolo nos queda ver que es PCCV_HE. Sin embargo, es muy sencillo, tenemos los inputs para el simulador S que son $x \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$, n y s y un reto e. Entonces podemos elegir de forma aleatoria $z \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y construir a como $a = \mathcal{E}_{DJN_s}(0; n, z)x^{-e} \mod n^{s+1}$, de forma que obtenemos una conversación (a, e, z) aceptada. La distribución de probabilidad es la misma, simplemente hemos cambiado la elección de elemento aleatorio de r a z. Ambos se eligen en $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ por lo que la distribución se mantiene entre la real y la simulada.

Juntando todo lo explicado vemos que el Σ-protocolo es válido. La demostración de solvencia especial y PCCV_HE son muy parecidas en todos los demás protocolos así que no entraremos en tanto detalle en los sucesivos casos.

Este protocolo se usará como paso intermedio en los demás protocolos más complejos,

no aparece *per se* en ninguno de los sistemas de voto.

Uno de entre dos candidatos es un cifrado del 0

Este es el protocolo que se usará en las elecciones de sí/no para asegurar que el voto emitido es alguna de las dos posibilidades. Si el votante puede votar m_1 o m_2, y su voto ha sido c una vez cifrado. Entonces $c = (1+n)^{m_i} r^{n^s}$ y la persona puede conseguir $x_1 = c(1+n)^{-m_1}$ o $x_2 = c(1+n)^{m_2}$, de los cuales puede asegurar que uno es un cifrado del 0. Solo el votante sabe de cual y para el otro usará un simulador S para conseguir una conversación que se acepte. Asumamos sin pérdida de generalidad que ha votado m_1 y entonces sabe que x_1 es un cifrado del 0, conoce ω tal que $x_1 = \mathcal{E}_{DJN_s}(0; n, \omega)$, sería el r aleatorio con el que cifró.

Partimos de lo inputs en común de P y V: $x_1, x_2 \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$, n y s.

El testigo del secreto es $\omega \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y el secreto que P conoce y quiere probar es que $x_1 = \mathcal{E}_{DJN_s}(0; n, \omega)$, la conversación quedaría:

1. $P \to V$:
   - *a*) P elige de forma aleatoria $r_1 \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$.
   - *b*) P elige un natural aleatorio de t bits e_2 con el que va a generar la conversación para x_2.
   - *c*) P invoca a S con el input n, x_2, e_2 para obtener la conversación (a_2, e_2, z_2).
   - *d*) P envía a_2 y $a_1 = \mathcal{E}_{DJN_s}(0; n, r_1)$ a V.
2. $V \to P$: V elige un número aleatorio $\tilde{\mathrm{e}}$ de t bits y se lo envía a P.
3. $P \to V$:
   - *a*) P calcula $e_1 = \tilde{\mathrm{e}} - e_2 \mod 2^t$.
   - *b*) P calcula $z_1 = r_1 \omega^{e_1} \mod n$.
   - *c*) P envía e_1, e_2, z_1, z_2 a V.

La verificación por parte de V se realiza de la siguiente forma: primero comprueba que x_1, x_2, a_1, a_2, z_1 y z_2 son relativamente primos con n. A continuación, verifica que $\tilde{\mathrm{e}} = e_1 + e_2 \mod 2^t$, $\mathcal{E}_{DJN_s}(0; n, z_1) \equiv a_1 x_1^{e_1} \mod n^s$ y $\mathcal{E}_{DJN_s}(0; n, z_2) \equiv a_2 x_2^{e_2} \mod n^s$.

Lo primero es obvio por la conversación, ya que como $e_1 = \tilde{\mathrm{e}} - e_2$ se tiene que $\tilde{\mathrm{e}} = e_1 + e_2$. La siguiente comprobación es igual que la de antes del cifrado del cero:

$$\mathcal{E}_{DJN_s}(0; n, z_1) = \mathcal{E}_{DJN_s}(0; n, r_1\omega^{e_1}) = \mathcal{E}_{DJN_s}(0; n, r_1)\mathcal{E}_{DJN_s}(0; n, \omega)^{e_1} = a_1 x_1^{e_1}$$

La última comprobación también es obvia si recordamos como funcionaba S. Este S simula una conversación con unos inputs comunes que puede ser aceptada por un verificador. En este caso se le dan los inputs n, x_2 y e_2 y el simulador va a generar aleatoriamente $z_2 \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y calcular a_2 de forma que la conversación se acepte. Por tanto $a_2 = \mathcal{E}_{DJN_s}(0; n, z_2) x_2^{e_2}$, así que es obvio que al verificar va a cumplirse.

Pero, si con S siempre se puede crear una conversación que se acepte, ¿cómo podemos estar seguros que este protocolo funciona y no están ambas creadas con S? Al tener que mandar a_1 antes a V y no conocer e_1 no se puede generar la conversación igual que se hace con el que no se conoce. Pero, ¿y si genera la conversación entera antes y luego manda el a_1 que le sale? En este caso la comprobación $\tilde{\mathrm{e}} = e_1 + e_2 \bmod 2^t$ lo delataría porque no puede cambiar ninguno para que la suma funcione y que las siguientes comprobaciones sigan saliendo. Entonces es claro que el protocolo funciona y garantiza que la persona intentando demostrar que conoce el secreto realmente lo conoce. Por lo tanto, por como se han calculado x_1 y x_2 tuvo que votar o m_1 o m_2 con lo que el voto es válido.

La demostración de la solvencia especial y sobre la prueba de ser PCCV_HE son muy similares y usan estrategias ya vistas en la primera. Se pueden consultar en [12].

Este protocolo se usa en las votaciones de sí/no para que un votante pueda demostrar que su voto es válido. A la vez que el voto cifrado, el individuo tiene que adjuntar al espacio público con los votos una conversación para que cualquier verificador pueda comprobar la validez de su voto.

Uno de entre L candidatos es un cifrado del 0

En este caso el problema es similar y la solución también lo es. El votante tiene que demostrar que su voto es uno de los posibles L. Entonces se calcularan:

$$\begin{aligned} x_1 &= c(1+n)^{-m_1} \\ x_2 &= c(1+n)^{-m_2} \\ &\vdots \\ x_L &= c(1+n)^{-m_L} \end{aligned}$$

Y se quiere demostrar que uno de ellos es un cifrado del 0. El votante P sabe cual, por tanto también conoce el $r \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ con el que se cifró, y ese es el testigo del secreto que quiere probar $\omega = r$. Supongamos sin pérdida de generalidad que ha votado m_1, por tanto $c = \mathscr{E}_{DJN_s}(m_1; n, r)$ y $x_1 = \mathscr{E}_{DJN_s}(0; n, r)$. El Σ-protocolo va a consistir en lo mismo que antes, se va a usar S para simular las conversaciones de los $L-1$ inputs restantes.

Partimos de lo inputs en común de P y V: $x_1, x_2, \ldots, x_L \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$, n y s.

El testigo del secreto es $\omega \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y el secreto que P conoce y quiere probar es que $x_1 = \mathscr{E}_{DJN_s}(0; n, \omega)$, la conversación quedaría:

1. $P \to V$:
   - *a*) P elige de forma aleatoria $r_1 \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$.
   - *b*) Para cada $i \in \{2, \ldots, L\}$:
     1) P elige un natural aleatorio de t bits e_i con el que va a generar la conversación para x_i.

2) P invoca a S con el input n, x_i, e_i para obtener la conversación (a_i, e_i, z_i).

c) P envía $a_2, \ldots, a_L$ y $a_1 = \mathcal{E}_{DJN_s}(0; n, r_1)$ a V.

2. $V \to P$: V elige un número aleatorio $\tilde{\mathrm{e}}$ de t bits y se lo envía a P.

3. $P \to V$:

a) P calcula $e_1 = \tilde{\mathrm{e}} - \left(\sum_{i=2}^{L} e_i\right) \bmod 2^t$.

b) P calcula $z_1 = r_1 \omega^{e_1} \bmod n$.

c) P envía $e_1, \ldots, e_L, z_1, \ldots, z_L$ a V.

La verificación por parte de V se realiza de la siguiente forma: primero comprueba que $x_1, \ldots, x_L, a_1, \ldots, a_L, z_1, \ldots, z_L$ son relativamente primos con n. A continuación, verifica que $\tilde{\mathrm{e}} = \sum_{i=1}^{L} e_i \bmod 2^t$, $\mathcal{E}_{DJN_s}(0; n, z_i) \equiv a_i x_i^{e_i} \bmod n^s$ para todo $i \in \{1, \ldots, L\}$.

La verificación de que funciona es exactamente igual que la del protocolo anterior, así como la prueba de la solvencia especial y la prueba de que es PCCV_HE.

Se usa en las votaciones de 1 de entre L candidatos y las de k entre L candidatos para probar que el voto emitido es válido. Junto con su voto cifrado la persona tiene que adjuntar una conversación que cualquier verificador pueda comprobar en cualquier momento.

La relación $ab \equiv c \bmod n^s$ entre textos en claro

En las votaciones en las que se debe elegir a lo sumo k candidatos entre L necesitamos un protocolo, que vamos a presentar ahora, el cual es necesario para poder asegurarse de que un votante no ha votado más de una vez al mismo candidato. Pero para hacer esto necesitamos poder disponer de una prueba de conocimiento, que verifique que el producto de dos textos en claro es igual a otro texto en claro, es decir dados $a, b, c \in \mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ se tiene que $ab \equiv c \bmod n^s$.

Partimos de los inputs en común, en este caso tendremos $x_a = \mathcal{E}_{DJN_s}(a; n, r_a)$, $x_b = \mathcal{E}_{DJN_s}(b; n, r_b)$ y $x_c = \mathcal{E}_{DJN_s}(c; n, r_c)$ elementos de $\left(\mathbb{Z}/\mathrm{n^{s+1}}\mathbb{Z}\right)^*$ y los enteros n y s.

El secreto que queremos verificar con este protocolo tiene muchos testigos. Sean los testigos $a, b, c \in \mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ y r_a, r_b y $r_c \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ tales que $ab \equiv c \bmod n^s$ y se mantengan las relaciones anteriores. En este caso hay muchos testigos porque P no solamente quiere probar que los textos cumplen la relación $ab \equiv c \bmod n^s$, también que las x_i son los textos cifrados de esos textos en claro. La conversación quedaría como sigue:

1. $P \to V$:

a) P elige aleatoriamente un $d \in \mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ y $r_d, r_{bd} \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$.

b) P calcula $\tilde{a}_1 = x_d = \mathcal{E}_{DJN_s}(d; n, r_d)$.

c) P calcula $\tilde{a}_2 = x_{bd} = \mathcal{E}_{DJN_s}(db; n, r_{db})$.

d) P manda $\tilde{a}_1$ y $\tilde{a}_2$ a V.

2. $V \to P$: V elige un número aleatorio de t bits e y se lo manda a P.

3. $P \to V$:

   a) P usa la propiedad homomorfica del cifrado para calcular los valores que dan de texto cifrado $x_a^e \tilde{a}_1$:

   $$\begin{aligned} x_a^e \tilde{a}_1 &= x_a^e x_d = \mathcal{E}_{DJN_s}(a; n, r_a)^e \mathcal{E}_{DJN_s}(d, r_d) = \mathcal{E}_{DJN_s}(ae + d \bmod n^s; n, r_a^e r_d \bmod n) \\ &= \mathcal{E}_{DJN_s}(z_1; n, z_2) \end{aligned}$$

   y después manda a V $z_1 = ae + d$ y $z_2 = r_a^e r_d$, pues P conoce todos los parámetros.

   b) P usa la propiedad homomórfica del cifrado para calcular los valores que dan de texto cifrado $x_b^{z_1}(\tilde{a}_2 x_c^e)^{-1}$:

   $$\begin{aligned} x_b^{z_1}(\tilde{a}_2 x_c^e)^{-1} &= x_b^{z_1}(x_{bd} x_c^e)^{-1} = \mathcal{E}_{DJN_s}(b; n, r_b)^{ea+d}(\mathcal{E}_{DJN_s}(bd; n, r_{bd})\mathcal{E}_{DJN_s}(c; n, r_c)^e)^{-1} \\ &= \mathcal{E}_{DJN_s}(bea + db - bd - ce \bmod n^s; n, r_b^{z_1}(r_{bd} r_c^e)^{-1} \bmod n) \\ &= \mathcal{E}_{DJN_s}(0; n, z_3) \end{aligned}$$

   y después manda a V $z_3 = r_b^{z_1}(r_{bd} r_c^e)^{-1} \bmod n$.

La verificación por parte de V empieza por comprobar que $\tilde{a}_1, \tilde{a}_2, z_1, z_2$ y z_3 son relativamente primos con n. A continuación comprueba que: $\mathcal{E}_{DJN_s}(z_1; n, z_2) = x_a^e \tilde{a}_1 \bmod n^{s+1}$ y $\mathcal{E}_{DJN_s}(0; n, z_3) = x_b^{z_1}(\tilde{a}_2 x_c^e)^{-1} \bmod n^{s+1}$.

P mediante el cálculo de z_2 y z_3 prueba que conoce r_a, r_b y r_c; ya que si no los conociera y estuviera cifrando los mensajes de la segunda parte con elementos aleatorios de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ no se correspondería en la comprobación en la fase de verificación, salvo con una probabilidad ínfima. Además con el cálculo de z_1 y con la variable de cifrado de la segunda parte se puede ver lo siguiente:

$$bz_1 - (db + e(c)) \equiv b(ea + d) - (db + ce) \equiv eab - ec \equiv 0 \bmod n^s \Leftrightarrow ab \equiv c \bmod n^s$$

Luego si la comprobación se verifica cifrando 0 con z_3 significa que a, b y c cumplen $ab \equiv c \bmod n^s$ que es el secreto que P quería demostrar que conocía.

La prueba de la solvencia especial en este caso no es similar a las anteriores, sin embargo es muy sencilla. Partamos de dos conversaciones aceptadas $(\tilde{a}_1, \tilde{a}_2, e, z_1, z_2, z_3)$ y $(\tilde{a}_1, \tilde{a}_2, \tilde{\mathrm{e}}, \tilde{\mathrm{z}}_1, \tilde{\mathrm{z}}_2, \tilde{\mathrm{z}}_3)$ con $\tilde{\mathrm{e}} \neq e$. Como ambas conversaciones son aceptadas cumplen que:

$$\left.\begin{aligned} z_1 b - db - ec &\equiv 0 \bmod n^s \\ \tilde{\mathrm{z}}_1\, b - db - \tilde{\mathrm{e}}\, c &\equiv 0 \bmod n^s \end{aligned}\right\} (z_1 - \tilde{\mathrm{z}}_1)b \equiv (e - \tilde{\mathrm{e}})c \bmod n^s$$

$$\begin{aligned} &\Rightarrow (ea + d - (\tilde{\mathrm{e}}\, a + d))b \equiv (e - \tilde{\mathrm{e}})c \bmod n^s \\ &\Rightarrow (e - \tilde{\mathrm{e}})ab \equiv (e - \tilde{\mathrm{e}})c \bmod n^s \end{aligned}$$

Esas operaciones se pueden realizar usando los inputs comunes y son las que tendrán

lugar en el exponente de $(1+n)$. Es decir se pueden obtener como $x_b^{z_1}\tilde{a_2}^{-1}x_c^{-e}$ con lo que solo usaríamos las variables de las conversaciones. Como se tiene que $e, \tilde{e} < 2^t < p, q$ luego mcd$((e-\tilde{e}), n) = 1$ y por tanto es invertible en $\mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ y al simplificar obtenemos la relación $ab \equiv c \mod n^s$. Luego de ambas conversaciones aceptadas se saca que se cumple $ab \equiv c \mod n^s$ que es el secreto que quería probar P.

Solo falta verificar que es PCCV_HE para tener que es un Σ-protocolo completo. Vamos a ver como funcionaría el simulador S partiendo de los input comunes de antes, $a, b, c \in \mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ y r_a, r_b y $r_c \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$, y un reto e. El simulador va a elegir aleatoriamente $z_1 \in \mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ y $z_2, z_3 \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y sea:

$$\tilde{a}_1 = \mathcal{E}_{DJN_s}(z_1, z_2)(x_a^e)^{-1}$$
$$\tilde{a}_2 = x_b^{z_1}(x_c^e\mathcal{E}_{DJN_s}(0, z_3))^{-1}$$

Por construcción, esta conversación será aceptada y además tiene la misma distribución de probabilidad que la conversación sin simular. Está tomando un elemento aleatorio de $\mathbb{Z}/\mathrm{n^s}\mathbb{Z}$ y dos de $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$, lo único que cambia es cuales son en cada caso, pero el conjunto es el mismo luego es un simulador válido. La conversación $(\tilde{a}_1, \tilde{a}_2, e, z_1, z_2, z_3)$ tiene la misma probabilidad que una conversación real.

Estos protocolos y pruebas van incluidos en las votaciones en las que se eligen k de entre L candidatos y se utiliza para asegurarse de que ningún voto está duplicado, es decir que no se votó dos veces a la misma persona. Como el sistema de este tipo de votaciones es ligeramente más complejo que los otros en los que solamente había que añadir la prueba vamos a explicarlo más en detenimiento.

La fase de preparación se mantiene idéntica, la situación es la misma y se tienen que generar las claves del cifrado DJN_s.

Al comenzar la fase de votación hay una primera parte en la que se procede de uno en uno con los votos. El votante elige de entre los $L + K - 1$ posibles candidatos, contando los ficticios para votos en blanco, y genera el $r \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ correspondiente para cifrarlo y lo guarda en su unidad privada. Una vez cifrado c, a continuación hay que calcular $x_i = c(1+n)^{-m_i}$ para todo $i \in \{0, \ldots, L+K-2\}$. Con estas x_i, la n y las s ya tenemos los inputs comunes para la prueba. Generalmente se usan pruebas de no interacción, donde el papel de verificador es sustituido por una función hash pública para todos. Con esto tenemos la primera parte en la que los votos están cifrados y se ha asegurado que son votos válidos.

Ahora viene la segunda parte, en esta nos tenemos que asegurar de que ninguno de los votos está repetido. Supongamos que tenemos m_{i_k} y m_{i_t} dos votos del votante i-ésimo con $k \neq t$. Como supuestamente deberían ser votos diferentes, $m_{i_k} - m_{i_t} \neq 0$ para todo par de votos. Entonces, podemos calcular el inverso de dicha diferencia $(m_{i_k} - m_{i_t})^{-1}$, es decir un número que cumple $(m_{i_k} - m_{i_t})(m_{i_k} - m_{i_t})^{-1} \equiv 1 \mod n^s$. Como podemos observar, la situación es casi la misma que la del Σ-protocolo que acabamos de explicar. La diferencia es que en este caso sí tenemos información sobre los textos en

claro, y lo que se busca es probar que esa relación la cumplen, no unos textos aleatorios que conoce el votante, sino unos concretos asociados al par de votos que se están comprobando. Es decir que cuando se dan los inputs comunes el verificador tiene que asegurarse de que son los adecuados. El verificador conoce de primeras el texto cifrado que corresponde a $(m_{i_k} - m_{i_t})$ que es $c_{i_k}(c_{i_t})^{-1}$ siendo $c_{i_k} = \mathcal{E}_{DJN_s}(m_{i_k}; n, r_{i_k})$ y $c_{i_t} = \mathcal{E}_{DJN_s}(m_{i_t}; n, r_{i_t})$, luego el primer input común es $x_a = c_{i_k}(c_{i_t})^{-1}$. Otro de los inputs comunes es $x_b = \mathcal{E}_{DJN_s}((m_{i_k} - m_{i_t})^{-1}; n, r_b)$, este es calculado por el votante con un $r_b \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ aleatorio, el verificador no tiene que comprobar este. El último de los inputs comunes tiene una complicación, no necesitamos 1, necesitamos un cifrado de 1 que el votante conozca y que el verificador pueda comprobar. Hay dos estrategias principales para este propósito: la primera es que las autoridades publiquen un cifrado de 1 como referencia para que los votantes puedan crear la conversación del Σ-protocolo, sin embargo, las autoridades deberán hacer público también el $r \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ con el que se ha cifrado porque es uno de los inputs privados y es necesario. Esto compromete la seguridad del mismo porque uno de los inputs privados es público. La segunda, y por la que se opta generalmente, es que el votante genere un cifrado de 1 de forma local y guarde la variable de cifrado $r_c \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ para el protocolo. Adicionalmente en este caso tiene que demostrar que $x_c = \mathcal{E}_{DJN_s}(1; n, r_c)$ es realmente un cifrado de 1. El votante deberá generar una prueba de cifrado del cero para $x_c(1+n)^{-1}$, de esta forma el verificador puede comprobar que es un cifrado de 1.

Vamos entonces a explicar la segunda parte de la fase de votación, recordemos que todos los votos son válidos. Ahora lo que tenemos que hacer es coger los votos de dos en dos, las $\binom{K}{2}$ posibles parejas de votos, y para cada una generar la conversación que permita verificar la relación con los inputs comunes $x_a = c_{i_k}(c_{i_t})^{-1}$, $x_b = \mathcal{E}_{DJN_s}((m_{i_k} - m_{i_t})^{-1}; n, r_b)$ y $x_c = \mathcal{E}_{DJN_s}(1; n, r_c)$ y con los inputs privados $a = (m_{i_k} - m_{i_t})$, $b = (m_{i_k} - m_{i_t})^{-1}$, $c = 1$ y las $r_a = r_{i_k} r_{i_t}^{-1}$, r_b y r_c. Una vez se tenga esta conversación hará falta también la conversación que permita verificar que $x_c(1+n)^{-1}$ es un cifrado de 0, la cual se puede adjuntar con cada una de las anteriores, o simplemente hacerla una vez para usar en todas y adjuntarla por separado.

El votante con los votos cifrados publica todas las pruebas de verificación de los votos asociados de alguna manera a cada uno. Además sube la prueba de la relación de multiplicación para cada pareja y la prueba de que x_c es un cifrado de 1. Así, cualquier verificador puede comprobar que los votos son válidos y que no se ha votado dos veces al miso candidato. Si fuera el caso $m_{i_k} - m_{i_t} \equiv 0 \mod n^s$ con lo que no se podría invertir y no pasaría la prueba de verificación y se anularían los votos.

5.4. Descifrado con umbral

Lo último de lo que vamos a hablar en esta sección es de que podemos hacer en el caso en el que haya autoridades corruptas. Evidentemente no se pude confiar ciegamente en la gente, independientemente de que haya sido elegida para realizar el recuento de la

votación, pero sí podemos confiar en que una fracción mayoritaria de la misma es honesta. Entonces, la solución que parece más obvia es forzar a que el descifrado se realice entre varias de las autoridades. Se cambiará el descifrado por uno que se llama descifrado con umbral. El cifrado es el mismo. La modificación que se hace para transformar el criptosistema es un reparto de la clave privada, el secreto, entre todas las autoridades de forma que se tengan que juntar al menos H de las A autoridades totales para descifrar los mensajes. Se puede ver esta estrategia tanto en [12] como en [6], básicamente implementaremos el esquema de Shamir que ya ha sido explicado en 2.2.1.

Para poder hacer esto necesitamos hacer operaciones propias de un cuerpo, vamos a necesitar sumar y restar y hacer exponenciaciones. El primero en apicar este esquema de reparto de secretos a criptografía fue Victor Shoup [14] y lo ideo para RSA. Como nosotros estamos usando el cifrado de Paillier el texto cifrado se encuentra en un espacio diferente. En su texto original el texto cifrado está en $(\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$, el nuestro está en $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$. Para garantizar que el método sea operativo lo primero que necesitamos es una restricción sobre p y q, los primos de los cuales n es producto. Es necesario que cumplan que $p' = \frac{p-1}{2}$ y $q' = \frac{q-1}{2}$ sean también primos. Se define el grupo de elementos que son cuadrados módulo n^{s+1}, $Q_{n^{s+1}} = \{c \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^* | c = z^2;\ z \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*\}$, el orden de este grupo es $p'q'n^s = \tilde{n}n^s$ con $\tilde{n} = p'q'$. Es decir, haremos las operaciones de grupo, sumar, multiplicar... en $Q_{n^{s+1}}$ y las de exponenciación, elevar un número a otro, en $\mathbb{Z}/\tilde{n}n^s\mathbb{Z}$. Para poder aplicar la interpolación, $\mathbb{Z}/\tilde{n}n^s\mathbb{Z}$ debería ser un cuerpo (que no lo es) o al menos que para todo i, j, el elemento $(i-j)$ tuviera inverso. No obstante, aunque lo tuviera, no se podría calcular pues no se conoce el valor de $\lambda(n)$ y por tanto se desconoce el orden del grupo $(\mathbb{Z}/\tilde{n}n^s\mathbb{Z})^*$. Para solucionar esto vamos a multiplicar (2.6) por un factor de forma que se puedan simplificar todos los denominadores. $\prod_{j=1\wedge i\neq j}^{H}(i-j)$ divide a $(H-i)!i!$ porque la diferencia más grande que puede haber es $(H-i)$, para $j > i$ las diferencias van de 1 a $(H-i)$, y además solo se pueden repetir números entre 0 e i, pues para $j < i$ tenemos que las diferencias van de 1 a i. Pero además, si nos fijamos en los números combinatorios podemos afirmar que $(H-i)!i!|H!$ ya que $\binom{H}{i} = \frac{H!}{(H-i)!i!} \in \mathbb{N}$, de modo que vamos a multiplicar todo por $H!$ para evitar calcular inversos.

$$H!d = H!f(0) = H!\sum_{i=1}^{H} d_i \prod_{j=1\wedge j\neq i}^{H} \frac{-j}{i-j} = \sum_{i=1}^{H} d_i L_{H,i}^*(0) \bmod \tilde{n}n^s$$

Con $L_{H,i}^* = H! \prod_{j=1\wedge j\neq i}^{H} \frac{-j}{i-j} \bmod \tilde{n}n^s$.

En el proceso de descifrado cada una de las autoridades honestas va a tener que generar su participación de los votos emitidos, para calcular el voto mediante el reparto de secretos. Para ello, lo primero que realizará será calcular, $c_i = c^{2d_iH!}$ (a d_i para la interpolación y obtener d, a 2 para asegurarnos de que es un elemento de $Q_{n^{s+1}}$ y a $H!$ para asegurarnos de que el proceso se mantiene en el grupo). Aquí es donde entra en juego el Σ-protocolo de igualdad de logaritmos discretos, para asegurarnos de que las autoridades elevan el texto cifrado a $d_iH!$.

Igualdad de logaritmos discretos

Este Σ-protocolo sirve para verificar que dos números x_1 y x_2 son potencias u-ésimas. Es decir que $x_1 = y_1^u$ y que $x_2 = y_2^u$. Partimos de los inputs comunes x_1, x_2, y_1 e $y_2 \in Q_{n^{s+1}} = \{c \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s+1}}\mathbb{Z}\right)^* | c = z^2;\ z \in \left(\mathbb{Z}/\mathrm{n}^{\mathrm{s+1}}\mathbb{Z}\right)^*\}$. El secreto que quiere demostrar P es que conoce u, es decir el logaritmo discreto común de x_1, x_2 en base y_1, y_2 respectivamente. Este u será el testigo del secreto y lo denotaremos ω, le fijaremos una longitud máxima de $(s+1)k$ bits donde k es la longitud de n. La conversación quedaría así:

1. $P \to V$:
   - *a*) P elige un número aleatorio r de longitud $(s+2)k+t$ bits, siendo t el parámetro de seguridad prefijado.
   - *b*) P calcula $a_i = y_i^r \bmod n^{s+1}$ para $i \in \{1,2\}$ y manda (a_1, a_2) a V.
2. $V \to P$: elige un número aleatorio e de longitud t bits y se lo manda a P.
3. $P \to V$: P calcula $z = r + e\omega$ y le manda z a V

En la fase de verificación V comprueba que $y_i^z \equiv a_i x_i^e \bmod n^{s+1}$ para $i \in \{1,2\}$. Esta comprobación funciona porque:

$$y_i^z \equiv y_i^{r+e\omega} \equiv y_i^r y_i^{e\omega} \equiv a_i (y_i^\omega)^e \equiv a_i x_i^e \bmod n^{s+1}$$

Ahora vamos a verificar la solvencia especial, supongamos que tenemos dos conversaciones aceptadas (a_1, a_2, e, z) y $(a_1, a_2, \tilde{\mathrm{e}}, \tilde{\mathrm{z}})$ las cuales como están aceptadas verifican: $y_i^z \equiv a_i x_i^e \bmod n^{s+1}$ y $y_i^{\tilde{z}} \equiv a_i x_i^{\tilde{e}} \bmod n^{s+1}$. Esto implica que si dividimos uno entre el otro nos queda $y_i^{z-\tilde{\mathrm{z}}} \equiv x_i^{e-\tilde{\mathrm{e}}} \bmod n^{s+1}$, es decir que:

$$log_{y_i}(x_i) = \frac{z - \tilde{\mathrm{z}}}{e - \tilde{\mathrm{e}}} \equiv \frac{r + e\omega - (r + \tilde{\mathrm{e}}\,\omega)}{e - \tilde{\mathrm{e}}} \equiv \frac{\omega(e - \tilde{\mathrm{e}})}{e - \tilde{\mathrm{e}}} \equiv \omega \bmod n^{s+1}$$

Luego permiten obtener el secreto.

Faltaría ahora verificar que es PCCV_HE, para ello como siempre crearemos un simulador S con los mismos inputs comunes de antes y un reto e y que tenga la misma distribución de probabilidad. Y de la misma forma que se hizo en casos anteriores se va a cambiar la aleatoriedad de r a z y como son elementos del mismo grupo que es $Q_{n^{s+1}}$ la distribución de probabilidad es la misma. Una vez tengamos el z se calculan las a_i de la siguiente manera $a_i = \frac{y_i^z}{x_i^e}$. De esta forma S ha simulado una conversación aceptada por cualquier verificador V.

Para usar esta prueba lo primero que necesitamos es un elemento al que llamaremos v que será una clave de verificación, a la que lo único que le pedimos es que genere $Q_{n^{s+1}}$, lo cual se puede tener eligiendo $v \in Q_{n^{s+1}}$ de forma aleatoria salvo una probabilidad despreciable. Entonces a cada autoridad se le será entregado $v_i = v^{d_i H!}$ para utilizar el Σ-protocolo de igualdad de logaritmos discretos con inputs comunes v_i, $c_i^2 = c^{4d_i H!}$, v y c^4, y siendo el secreto que quiere demostrar que c_i está realmente elevado a $d_i H!$. Como

no confiamos en la autoridad y no estamos seguros de que haya elevado c al cuadrado calculando c_i, lo hacemos en la prueba también para asegurarnos de trabajar en $Q_{n^{s+1}}$. Pettersen en [12] recomienda elevar al cuadrado también en la fase de combinación y en [6] Jurik hace lo mismo. En principio si el c_i vuelve con una prueba positiva de la igualdad de logaritmos discretos esto no sería necesario porque ya sabríamos que ha sido elevado al cuadrado, y en caso de regresar negativo se corta la computación. Sin embargo, si de alguna forma pudiera intervenir un adversario en el proceso intermedio es una medida cautelar razonable. Hagamos entonces lo siguiente:

$$c' \equiv \prod_{i=1}^{H} c_i^{2L^*_{H,i}} \equiv \prod_{i=1}^{H} c^{2H!d_i \cdot 2H! \sum_{j=1 \wedge j \neq i}^{H} \frac{-j}{i-j}} \equiv c^{4(H!)^2 \sum_{i=1}^{H} d_i \sum_{j=1 \wedge j \neq i}^{H} \frac{-j}{i-j}} \equiv c^{4(H!)^2 d} \bmod n^{s+1}$$

El c ha sido encriptado de la forma habitual del cifrado DJN_s y d cumple $d \bmod n \in (\mathbb{Z}/\mathrm{n}\mathbb{Z})^*$ y $d \equiv 0 \bmod \lambda$. A este d le vamos a añadir una condición más para poder descifrar sin conocerlo, vamos a pedirle que $d \equiv 1 \bmod n^s$. Además si no cumple esto el criptosistema falla completamente porque se podría obtener d conociendo el resultado de la votación. De esta forma siguiendo con lo de arriba lo que nos queda es:

$$c^{4(H!)^2 d} \equiv (n+1)^{4(H!)^2 d m} r^{n^s 4(H!)^2 d} \equiv (n+1)^{4(H!)^2 m} \bmod n^{s+1}$$

La r desaparece por el teorema de Carmichael 2.4 y porque d es un múltiplo de λ. La d desparece del numerador en $(n+1)$ porque tiene orden n^s en $\left(\mathbb{Z}/\mathrm{n}^{\mathrm{s}+1}\mathbb{Z}\right)^*$ y $d \equiv 1 + n^s k$. Ahora haciendo uso del algoritmo 4 se puede conseguir $4(H!)^2 m \bmod n^s$, con lo cual para conseguir el mensaje original, y con él el recuento de los votos, se tiene que calcular $\left(4(H!)^2\right)^{-1} \bmod n^s$ y multiplicar ambas cosas. En este método de descifrado el único que conoce la clave privada es el que la creó en la fase de generación de llaves, las autoridades no necesitan saberla para descifrar. Así tenemos garantizado que una autoridad corrupta no va a conocer el secreto, y como necesita colaborar para usarla es muy difícil que convenza a otras para descifrar lo que quiere averiguar.

Capítulo 6

Conclusiones

El estudio realizado en este trabajo ha permitido profundizar en el cifrado de Paillier y explorar sus aplicaciones en el ámbito del voto electrónico. Destacando su potencial para construir sistemas seguros, con buenas propiedades y respetuosos con la privacidad de los usuarios.

En primer lugar, hemos presentado unos preliminares necesarios para entender el funcionamiento del cifrado de Paillier. Estos incluyen conceptos básicos de álgebra como la función φ de Euler y el teorema chino de los restos, y otros un poco menos habituales como la función de Carmichael. Además se hace una breve introducción a conceptos básicos de criptografía y reparto de secretos, dada su aplicabilidad al descifrado umbral. Al final encontramos un problema de optimización el cual es necesario para el recuento en votaciones de tipo 1 entre L y k entre L candidatos.

A continuación presentamos el cifrado de Paillier, en la sección hablamos tanto de cómo generar las claves como de cifrado y descifrado. En concreto, es muy importante verificar la propiedad homomórfica de la función de cifrado para su aplicabilidad al voto electrónico, y verificar la biyectividad para garantizar que el descifrado nos lleve al mensaje original. Además, hay varios puntos en los que se explican las posibles debilidades del cifrado. Este cifrado se rompe si se puede encontrar el valor de la función de Carmichael para un entero grande del que no se conoce su factorización. Sin embargo, hay problemas, en principio más difíciles, que también pueden romper el cifrado, el Fact[n] y el RSA[n,n], y hay un problema equivalente al D-Class[n] que es el CR[n], es decir, que el cifrado, aunque esté protegido por un problema en principio intratable esto se sustenta en que no se rompa Fact[n], RSA[n,n] ni CR[n].

El análisis de las modificaciones propuestas por Damgård, Jurik y Nielsen ha mostrado que es posible extender la funcionalidad del esquema original para soportar mensajes más largos, reducir el factor de expansión y mantener las propiedades homomórficas. Estas mejoras resultan especialmente útiles en escenarios de votaciones complejas, donde los participantes pueden elegir entre múltiples opciones o seleccionar subconjuntos de candidatos. La sección va acompañada de pruebas respecto a la seguridad y a la conservación de las propiedades de Paillier.

La parte final del trabajo es la que está dedicada al voto electrónico, no perdamos de vista que uno de los objetivos del trabajo era aplicar estos cifrados a ello. El voto electrónico, igual que el clásico, consta de tres fases: preparación, votación y recuento. El papel del cifrado y los protocolos será garantizar que estas tengan lugar de forma íntegra y transparente, pero, a su vez, preservando la privacidad del votante.

Los esquemas de voto cuando asumimos honestidad por parte de los votantes y las autoridades, aunque sean curiosas, son muy sencillos. Un simple cifrado y un reparto de secretos con la clave es suficiente. En el momento en el que votantes deshonestos se empiezan a considerar es cuando surge la necesidad de probar conocimiento sin revelarlo, manteniéndolo secreto. Es aquí cuando aparecen las pruebas de conocimiento cero, serán muy útiles para que los votantes puedan garantizar que el voto emitido es válido. Tenemos las pruebas de *Cifrado del 0*, *Uno de entre dos candidatos es un cifrado del 0*, *Uno de entre L candidatos es un cifrado del 0* y *La relación* $ab \equiv c \mod n^s$ *entre textos en claro*. El último inconveniente a considerar, que las autoridades no sean honestas, se soluciona de la misma manera que la privacidad, forzando a que el descifrado se realice entre varias de las autoridades. Este descifrado umbral para garantizar que funciona requiere que los elementos estén en $Q_{n^{s+1}}$ y, aun así, como no es un cuerpo multiplicaremos el exponente por $H!$ para evitar el cálculo de inversos. De hecho necesitaremos otra prueba de conocimiento cero que es *Igualdad de logaritmos discretos* para asegurar que el descifrado umbral funciona bien.

En conclusión, el cifrado de Paillier, junto con sus variantes y protocolos asociados, proporciona una base sólida para el desarrollo de sistemas de votación electrónica modernos. Permiten garantizar todos los aspectos clave de una votación mejor que estándares habituales como RSA y son también de clave pública.

Anexo

En primer lugar no utilizaremos un algoritmo de generación de claves, como vamos a hacer pruebas más comedidas utilizaremos como primos $p = 5$ y $q = 7$ y $s = 4$ como variable en la versión de Damgård-Jurik-Nielsen

[1]:
```
p = 5
q = 7
s = 4
n = p*q
lamb = 12
```

Ahora vamos a ver la función de cifrado de Paillier

[2]:
```
def cifra(m, n, g=None):
    Zn = Zmod(n)
    Zn2 = Zmod(n**2)
    Zn2e = Zn2.unit_group()
    Zne = Zn.unit_group()

    if g is None:
        g = 1 + n

    g = Zn2(g)

    if not g.is_unit():
        return print('g no es válido')

    while True:
        r = Zn.random_element()
        if r.is_unit():
            break

    r = Zn2(r)

    m = Zn(m)
```

```
    c = Zn2(g^m*r^n)
    return c
```

Ahora vamos a realizar unas pruebas de cifrado, en primer lugar tenemos que elegir un mensaje, tomaremos $m_1 = 3$ y $m_2 = 13$. Ahora vamos a cifrar el mensaje varias veces para ver como cambia con cada cifrado, ejemplificando con esto que es un cifrado no determinista.

[3]:
```
m1 = 3
m2 = 13
c = cifra(m1, n)
c1 = cifra(m1, n)
c2 = cifra(m2, n)
c3 = cifra(m2, n)
print(c)
print(c1)
print(c2)
print(c3)
```

```
544
683
358
58
```

De hecho vamos a observar el total del conjunto de mensajes cifrados del $m_1 = 3$, pero para eso necesitaremos una ligera modificación de la función de cifrado:

[4]:
```
def cifrace(m, n, g=None):
    Zn = Zmod(n)
    Zn2 = Zmod(n**2)
    Zn2e = Zn2.unit_group()
    Zne = Zn.unit_group()

    if g is None:
        g = 1 + n

    g = Zn2(g)

    if not g.is_unit():
        return print('g no es válido')

    m = Zn(m)
    l = list()
```

```
    for r in Zn:
        if r.is_unit():
            r = Zn2(r)
            c = Zn2(g^m*r^n)
            l.append(c)
    return l

m1 = 3
lista = cifrace(m1, n)
print(lista)
```

```
[106, 683, 642, 44, 531, 792, 144, 681, 983, 1217, 781, 1083, 142, 444, 8,␣
↪242, 544, 1081, 433, 694, 1181, 583, 542, 1119]
```

Ahora vamos a ver el algoritmo de descifrado de Paillier.

[5]:
```
def L(a, lamb, n):
    Zn = Zmod(n)
    Zn2 = Zmod(n**2)
    a = Zn2(a)
    sol = (Integer(pow(a, lamb, n**2))-1)/(n)
    return Zn(sol)

def descifr(c, n, lamb, g=None):
    Zn = Zmod(n)
    if g is None:
        g = 1 + n
    m = Zn(L(c, lamb, n)/L(g, lamb, n))
    return m
```

Una vez tenemos el algoritmo de cifrado y de descifrado, podemos verificar que los mensajes de antes corresponden a los textos en claro que deberían.

[6]:
```
d = descifr(c, n, lamb)
d1 = descifr(c1, n, lamb)
d2 = descifr(c2, n, lamb)
d3 = descifr(c3, n, lamb)
print(d)
print(d1)
print(d2)
print(d3)
```

```
3
3
13
```

13

Otra cosa que podemos ejemplificar ahora es que el cifrado es homomórfico. Si multiplicamos $c \cdot c2$ al descifrar el valor obtenido nos debería de dar 16 que es la suma de ambos textos en claro.

[7]:
```
Zn2 = Zmod(n**2)
c_mul = Zn2(c*c2)
m_mul = descifr(c_mul, n, lamb)
print(c_mul)
print(m_mul)
```

1202
16

Ahora vamos a utilizar la versión de Damgård-Jurik-Nielsen y vamos a ver sus funciones de cifrado y descifrado.

[8]:
```
def cifraDJN(m, n, s, g=None):
    Zn = Zmod(n)
    Zns = Zmod(n**s)
    Znss = Zmod(n**(s+1))
    Znsse = Zn2.unit_group()
    Zne = Zn.unit_group()

    if g is None:
        g = 1 + n

    g = Znss(g)

    if not g.is_unit():
        return print('g no es válido')

    while True:
        r = Zn.random_element()
        if r.is_unit():
            break

    r = Znss(r)

    m = Zns(m)
    c = Znss(g^m*r^(n^s))
    return c

def LDJN(a, n, s):
```

```
    Zns = Zmod(n**(s-1))
    Znss = Zmod(n**s)
    sol = (Integer(Znss(a))-1)/(n)
    return Zns(sol)

def descifrDJN(c, n, s, lamb, g=None):
    Zns = Zmod(n**s)
    if g is None:
        g = 1 + n
    aux = Zns(recuexp(pow(c, lamb, n**(s+1)), n, s))
    m = Zns(aux)/Zns(lamb)
    return m

def recuexp(a, n, s):
    i = 0
    for j in range(1,s+1):
        t1 = LDJN(a, n, j+1)
        t2 = i
        for k in range(2,j+1):
            i = i-1
            Znj = Zmod(n**j)
            t2 = Znj(t2*i)
            t1 = Znj(t1 - (t2*n**(k-1)*inverse_mod(factorial(k), n**j)))
        i = Integer(t1)
    return i
```

Una vez vistas las funciones ahora vamos a hacer pruebas con ellas: en primer lugar, al igual que antes vamos a ver diferentes cifrados para un mismo mensaje que es $m = 325746$ y luego cifraremos también otro mensaje $m_2 = 999666$.

[9]:
```
Zns = Zmod(n**s)
m = 325746
m = Zns(m)
m2 = 999666
m2 = Zns(m2)
c = cifraDJN(m, n, s)
c1 = cifraDJN(m, n, s)
c2 = cifraDJN(m2, n, s)
print(c)
print(c1)
print(c2)
```

```
17925389
```

```
3399827
48755936
```

Ahora, al igual que antes, vamos a verificar que el descifrado funciona y ejemplificar la propiedad homomórfica que se mantiene con respecto al cifrado original de Paillier

[10]:
```
Znss = Zmod(n**(s+1))
mul = Znss(c*c2)
d = descifrDJN(c, n, s, lamb)
print(d)
d1 = descifrDJN(c1, n, s, lamb)
print(d1)
d2 = descifrDJN(c2, n, s, lamb)
print(d2)
dm = descifrDJN(mul, n, s, lamb)
print(dm)
325746+999666
```

```
325746
325746
999666
1325412
```

[10]:
```
1325412
```

Bibliografía

[1] María Arce Muela. Reparto de secretos. *Trabajo de Fin de Grado, Universidad de Cantabria*, octubre 2016.

[2] Zhengjun Cao and Lihua Liu. The paillier's cryptosystem and some variants revisited. *International Journal of Network Security*, 19(1):91–98, January 2017.

[3] Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q Nguyen. Paillier's cryptosystem revisited. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 206–214, 2001.

[4] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9:371–385, 2010.

[5] Thomas W Hungerford. *Algebra*, volume 73. Springer Science & Business Media, 2012.

[6] Mads Johan Jurik. *Extensions to the paillier cryptosystem with applications to cryptological protocols*. BRICS Shanghai, China, 2003.

[7] Neal Koblitz. *A course in number theory and cryptography*, volume 114. Springer Science & Business Media, 1994.

[8] Neal Koblitz. *Algebraic aspects of cryptography*, volume 3. Springer Science & Business Media, 2004.

[9] Helger Lipmaa. Secure electronic voting protocols. *The Handbook of Information Security*, 2, 2006.

[10] Joseph Lipman. Units mod p^n. https://www.math.purdue.edu/~jlipman/553/units-mod-p%5En.pdf, 2003. Lecture notes, Purdue University.

[11] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[12] Nina Pettersen. Applications of paillier's cryptosystem. Master's thesis, NTNU, 2016.

[13] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[14] Victor Shoup. Practical threshold signatures. In *International conference on the theory and applications of cryptographic techniques*, pages 207–220. Springer, 2000.

[15] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2003.