Universidad de Valladolid

Escuela de Ingeniería Informática
de Segovia

Trabajo Fin De Grado

Grado en Ingeniería Informática de Servicios y Aplicaciones

Compartición de Secretos y Aplicaciones

Autor: ***Iván Muñoz Martín***

Tutor: José Ignacio Farrán Martín

Agradecimientos

En primera instancia me gustaría dar las gracias a José Ignacio Farrán y a Diego Ruano por haber tutorizado mi Trabajo de Fin de Grado. Les agradezco principalmente el haberme transmitido los conocimientos que he podido adquirir durante el desarrollo del trabajo, así como su disposición a lo largo del final de esta etapa. También me gustaría mencionar la importancia del resto del profesorado de la Universidad de Valladolid, gracias a los cuales he podido estudiar y posteriormente profundizar en estas materias. En definitiva, gracias a todo el equipo de profesores que me han acompañado durante el grado, por haberme ayudado y apoyado a formarme en lo que será mi futuro profesional.

Resumen

En muchos sectores y ámbitos profesionales existe cierta información confidencial que no conviene exponer. Dado que se trata de un tipo de información cuyo acceso podría tener numerosas consecuencias, es necesario utilizar algún método que oculte dicho contenido. En este sentido, la criptografía se encarga de proteger estos datos al convertirlos en secuencias que resulten ilegibles para cualquier elemento que no tenga implicación directa en ellos.

El presente documento se centrará en dos aplicaciones basadas en los esquemas de compartición de secretos: la votación y subasta electrónica. Para entender correspondientemente estos protocolos, hemos empezado explicando distintas herramientas criptográficas, y los esquemas de compartición de secretos.

Dentro de los esquemas de compartición de secretos hemos basado nuestro estudio en el esquema de Shamir, los esquemas que pueden ser verificados públicamente y los esquemas de compromiso. Entre los esquemas de compartición de secretos verificables existen los esquemas que son verificados por una entidad verificadora (VSS) y los esquemas en los que cualquier participante puede verificar la exactitud de los datos transmitidos. Estos últimos, son denominados esquemas de compartición de secretos verificables públicamente $(PVSS)$.

Después trataremos el protocolo de computación segura entre múltiples partes (MPC). El protocolo MPC dotará de total seguridad a nuestros diseños de votación y subasta electrónica.

Para finalizar nos centraremos en la votación y subasta electrónica. En estas secciones se explica todos los conceptos correspondientes para entender bien cómo funcionan estos dos sistemas, siempre aportando ejemplos numéricos para entender a la perfección los algoritmos que lo componen.

Abstract

In many sectors and professional fields there is certain confidential information that should not be exposed. Since it is a type of information whose access could imply an numerous consequences, it is essential to use a method that hides this content. In this sense, cryptography is in charge of protecting this data by converting them into sequences that are unreadable for any element that does not have direct involvement in them.

This document will focus on protocols that are gaining importance with the passage of time, and that are based on secret sharing schemes. Some examples of these

protocols and on which we will base our documentation are voting and electronic auctioning. To understand these protocols correspondingly, we have started by explaining different cryptographic tools, and secret sharing schemes.

Within the secret sharing schemes we have based our study on Shamir's scheme, publicly verifiable schemes and commitment schemes. Among the verifiable secret sharing schemes there are schemes that are verified by a verifying entity (VSS) and schemes in which any participant can verify the correctness of the transmitted data. The latter are called Publicly Verifiable Secret Sharing Schemes $(PVSS)$.

Then we will deal with the multi-party coputation (MPC) protocol. The multi-party computation will provide total security to our designs of voting protocols and electronic auction.

Finally, we have focused our document on voting and electronic auctioning. In these sections, all the corresponding concepts are explained to understand how these two systems work, always giving numerical examples to fully understand the algorithms that compose it.

Índice general

Capítulo 1

Introducción

En todo protocolo criptográfico nuestro objetivo principal es asegurar la privacidad de la información. Se pretenderá plasmar la idea de que existe una serie de herramientas criptográficas que se encargan de dar consistencia a los mensajes. Entre estos algoritmos criptográficos se encuentran los esquemas de compartición de secretos, que como su propio nombre indica es una técnica criptográfica que consiste en dividir en particiones un secreto y repartir estas particiones entre diversos participantes. La división del mensaje implica que la facilidad de su reconstrucción se atribuye al conocimiento del mayor número de particiones en las que se dividió el secreto. De esta forma se espera que la obtención de dicho secreto solamente sea posible con el pleno conocimiento de todas las partes del secreto, es decir, o se tiene pleno conocimiento sobre el secreto o no se sabe nada sobre él.

Este tipo de esquemas sirven como almacén seguro de información. El secreto se encuentra distribuido en diferentes lugares, lo cual garantiza una mayor protección a la hora de recibir posibles ataques de agentes externos. Además, desde esta perspectiva, no sería suficiente con simplemente dividir la información, sino que se debe ofrecer garantías de protección de acceso a estos datos desde cualquier tipo de medio.

Para comprender mejor los esquemas de compartición de secretos, podemos entenderlos como una especie de rompecabezas. Las piezas del rompecabezas se reparten entre los distintos jugadores con el objetivo de garantizar la seguridad de la solución. Así, varios jugadores podrían tener piezas que por separado carecen de sentido, pero al juntarse se llegará a construir la figura en su plenitud.

La idea general es sencilla, vamos a suponer que nosotros queremos guardar el código de seguridad de nuestra tarjeta de crédito (compuesto por cuatro números). Si nosotros facilitamos cada dígito a cuatro amigos diferentes, entonces una persona ajena a esta información tendrá unas 10^4 combinaciones posibles para descifrarlo. La seguridad que hemos conseguido para que no se pueda detectar el número secreto se-

ría buena, pero si se juntan los tres amigos con sus respectivos dígitos confidenciales, la probabilidad que tienen de adivinar dicho número sería mayor, pues solo existirían 10 posibilidades para llegar a saber el código de seguridad de dicha tarjeta. Este ejemplo en un caso sencillo en el que solo se divide el secreto en cuatro unidades, pero es suficiente para observar cómo se complica la obtención del mensaje dependiendo de la información de la que se disponga.

Los precursores de los esquemas de compartición de secretos fueron Blackley y Shamir en el año 1979. Ambos trabajaron de manera independiente; Shamir basó su algoritmo en los polinomios interpoladores y Blacley en esquemas vectoriales. A pesar de enfocar sus algoritmos de distintas maneras, los dos procedían igual en el siguiente aspecto: si tenemos un número de particiones n del secreto S, fijando un número t, el cual corresponde con el mínimo número de particiones a partir del cual se podría acceder a la información. A este tipo de esquemas se les denominó esquemas $(t, n)-$umbrales, en los que si tenemos un número menor de t particiones no sabemos nada sobre el secreto. Sin embargo, si el número de particiones que tenemos en nuestro poder es mayor a t se tiene pleno conocimiento sobre el secreto.

El esquema de mayor popularidad dentro de los esquemas de compartición de secretos es el esquema de Shamir. Este consiste en dividir un secreto S entre n participantes, y se tiene como límite para reconstruir S t particiones. Si se juntan t particiones de los n participantes que contienen el secreto pueden reconstruir el secreto S, pero ningún subgrupo de $t-1$ participantes que contiene una partición del secreto podrá hacerlo. Para llevar a cabo esto se define el número t, y se generan aleatoriamente los coeficientes necesarios para construir un polinomio de grado $t-1$. El método para recuperar el secreto S, es reconstruir el polinomio $f(x)$ a partir de las particiones, esto se hace mediante la interpolación de Lagrange.

Entre los esquemas de compartición de secretos, también se centrará el estudio en los esquemas verificables (VSS) y los públicamente verificables $(PVSS)$, pues estos permitirán que todos los datos sean tramitados y verificados. Además, los esquemas de compromiso nos permitirán utilizar un valor que permanecerá oculto para más tarde ser revelado.

Los esquemas de compartición de secretos tienen diversas aplicaciones en el mundo real. Entre alguna de estas aplicaciones se encuentra: el control de accesos, la apertura de cajas de seguridad, la inicialización de dispositivos militares, y procesos electrónicos como la subasta y la votación electrónica. Nosotros hemos basado nuestro estudio en estas dos últimas aplicaciones, y la compartición de secretos unido con la computación segura entre múltiples partes (MPC), dotará a nuestros protocolos de transparencia, solidez y seguridad.

Pretenderemos destacar las ventajas en los protocolos de subasta y votación elec-

trónica frente a los sistemas tradicionales que todos conocemos. Se intentará asegurar la fiabilidad y eficiencia, motivo de debilidad que no les permite ser más relevantes que los sistemas tradicionales en la actualidad.

1.1. Motivación

Este trabajo consta de una descripción teórica de todos los conceptos matemáticos y criptográficos. Se presentará la compartición de secretos y el protocolo *MPC*, con el objetivo de ofrecer una mejor comprensión sobre cómo poder dotar de seguridad y al mismo tiempo, ofrecer una serie de pautas eficientes de cara a la creación de los protocolos de votación y subasta electrónica.

Se diferencian dos partes en el documento, la primera parte es una descripción teórica de todos los conceptos para entender mejor los protocolos de votación y de subasta electrónica. Siempre se ha tenido en cuenta los conceptos tanto matemáticos como criptográficos para comprender mejor esta primera parte, describiendo la teoría detrás de la compartición de secretos y del protocolo *MPC* que nos servirá para explicar los protocolos de votación y subasta electrónica de forma eficiente. En la segunda parte del documento hemos desarrollado todo el proceso de los protocolos de votación y subasta electrónica, y la justificación matemática que hay detrás de todos sus conceptos, siempre ejemplificando numéricamente para que se comprenda el funcionamiento y la importancia del proceso, añadiendo en pseudocódigo lo algoritmos que se presentan.

La votación y subasta electrónica se ha convertido en la actualidad en recursos muy utilizados. Cada vez son más las organizaciones que están buscando soluciones para estos protocolos. La necesidad de eliminar desplazamientos y dotar de sencillez al proceso, implica que proyectos blockchain como Cardano estén intentando lanzar un prototipo totalmente seguro de votación electrónica, y empresas tan potentes en la venta de artículos online como eBay se hayan lanzado ya a las subastas electrónicas de algunos de sus productos.

Para entender todos los conceptos desarrollados en la votación y subasta electrónica es necesario comprender previamente los esquemas de compartición de secretos. Los esquemas verificables (VSS) y los esquemas verificables $(PVSS)$, nos proporcionan la base para que todos los datos tanto de la votación como de la subasta electrónica sean verificados. En el caso de la subasta electrónica también tienen relevancia los esquemas de compromiso, que se desarrollan en la sección 4.4 ya que estos permitirán que el valor de las pujas realizadas por los licitadores no sean nunca revelados, salvo el valor de la puja ganadora que adquirirá el artículo.

Además es fundamental, por sus beneficios, un conocimiento integral tanto de la justificación como del proceso e implementación de los protocolos, aunque esto haya supuesto un mayor tiempo en el desarrollo del trabajo de investigación. Se pretende combinar los conocimientos adquiridos durante el grado con la investigación teórica llevada a cabo durante la revisión bibliográfica. En este sentido se tratará de llevar a cabo los objetivos que se proponen a continuación.

1.2. Objetivos

Ahora mostramos una serie de objetivos que se pretenden alcanzar tras la realización del presente trabajo fin de grado.

1. En un primer momento se ofrece un resumen tanto de los conceptos matemáticos como de las herramientas criptográficas básicas: Prueba de conocimiento cero, Problema del logaritmo discreto, Difiie-Hellman, Funciones hash y Fiat Shammir. Teniendo en cuenta la amplitud de la criptografía, buscaremos como objetivo poder seguir de forma más eficiente y con detalle las siguientes secciones de votación y subasta electrónica.

2. Realizar un seguimiento por distintas ramas de la criptografía hablando de la compartición secretos, detallando los esquemas de compartición de secretos verificables (VSS), los esquemas de compartición de secretos públicamente verificables $(PVSS)$ y los esquemas de compromiso. Estas explicaciones tendrán como objetivo principal la ejemplificación y las justificaciones matemáticas de todos los protocolos que se desarrollen posteriormente.

3. Como ya se expuso, siempre que existe información confidencial, esta puede ser atacada por adversarios ajenos. Por este motivo, es fundamental conocer las diversas formas de ataques y los distintos tipos de atacantes que existen para poder anticiparse a ellos. Con este objetivo, hablamos de la creación de protocolos fiables a través de la computación segura entre múltiples partes (MPC), de manera que esta sección nos servirá para explicar como agentes externos pueden tener el control de partes para intentar entorpecer el protocolo, y como con el uso de el protocolo MPC diseñaremos posteriormente protocolos totalmente seguros.

4. Aprender todos los conceptos del protocolo de votación electrónica. Este punto se encuentra dividido en tres partes diferenciadas, para entender y justificar el protocolo de la mejor manera. En la primera parte se tiene como objetivo explicar

de manera general el protocolo, para posteriormente, en la segunda parte estudiar los conceptos matemáticos que justifican el protocolo. Por último se observan las pruebas que se necesitan desarrollar con sus algoritmos en pseudocódigo correspondientes.

5. Estudiar la relevancia y funcionamiento del protocolo de subasta electrónica, utilizando los esquemas de compartición de secretos descritos. Con la utilización de estos esquemas y la *MPC* conseguiremos crear un diseño totalmente seguro.

Capítulo 2

Planificación del proyecto

En todo proyecto que se realice siempre hay que llevar a cabo una estructuración del trabajo que se va abordar. Nosotros hemos intentado dar en todo momento un punto de vista totalmente práctico en un estudio teórico. Se han realizado diseños de protocolos totalmente seguros utilizando toda la información teórica que se ha analizado.

Se pueden identificar durante la vida del proyecto etapas muy marcadas, que serán consecuentes entre sí. En la primera etapa se desarrolla un estudio general sin entrar en mayor divulgación ni explicación. Una segunda etapa de conceptos y justificaciones matemáticas y una última etapa de diseño en la que se crearán protocolos funcionales y fiables.

Existe una metodología sucesiva en la que todas las secciones del trabajo tendrán relevancia en posteriores puntos. Nosotros no podemos realizar la construcción completa de los protocolos sin haber entendido conceptos teóricos anteriores. Esta manera de afrontar los protocolos nos ha permitido identificar los puntos más importantes y centrar nuestros estudios en ellos para resolverlos eficientemente.

2.1. Estimación de costes

Antes de realizar un presupuesto adecuado del proyecto, hay que evaluar varios factores que intervienen en la estimación de costes:

- **Entorno**. Como herramienta externa empleamos **TexMaker**, editor de texto gratuito utilizado para la composición del presente documento. Este editor incluye soporte para más de 18 idiomas, corrección ortográfica, visor incorporado en pdf. La instalación y su posterior enmaquetado son factores a tener en cuenta a la hora de estimar un presupuesto.

- **Personal**. Para este proyecto se dispone de un único trabajador que será el encargado de ejecutar distintos roles:

  1. Jefe de proyecto. Se necesita que se realice una planificación y una selección de contenido. No es sencillo porque es necesario un proceso de búsqueda de toda la teoría que se podría abordar en el proyecto y pensar qué es lo más adecuado a tratar para enfocar el presente documento. El Jefe de proyecto es el encargado de realizar las correcciones del documento. Además, es el contacto directo en las reuniones con los tutores. Este rol desarrolla una estimación de las horas previa a la construcción del proyecto, siempre intentando tener en cuenta incrementos e imprevistos que pueden surgir.

  2. Analista. Después de una selección de contenido por parte del Jefe de proyecto, el analista es el encargado de realizar un estudio más exhaustivo sobre la información. Entre las funciones de este se encuentra redactar y enmaquetar el documento en su totalidad, así como detallar todas las partes teóricas. Este rol ocupará la mayor parte de tiempo estimado del proyecto.

  3. Diseñador. Una vez que se analizó toda la información y se realizó el estudio teórico por parte del analista, llega una fase de diseño. Este rol se encarga de la parte práctica del proyecto, que utiliza la información previa del analista para demostrar que con un previo estudio de los conceptos se puede llegar a diseñar protocolos eficientes y sólidos.

- **Horas trabajadas**. El tiempo que han estado involucrado los distintos perfiles en el proyecto, ha cambiado en función de los distintos meses y la cercanía a la finalización del mismo. Para los meses de octubre y noviembre del año 2020 al igual que para los meses de enero, febrero, marzo, abril, mayo, junio y julio del año 2021 el tiempo involucrado por el personal que forma parte del proyecto es de $5 - 7$ horas semanales. En el mes de septiembre y octubre del año 2021 se incrementaron notablemente la cantidad de horas empleadas ascendiendo hasta $24 - 30$ horas semanales, y el último mes de noviembre se han llegado a realizar $40 - 44$ horas semanales hasta terminar el proyecto.

  Para explicar este proceso de recuento de la cantidad de horas empleadas, podemos ver la tabla 2.1, en la que se ha tenido en cuenta los factores explicados anteriormente desde instalación del entorno, enmaquetado, revisiones, reuniones y presentación del proyecto.

Meses	Horas/Semana	Nº de semanas	Horas
Octubre- Junio	$5-7$	39	$195-273$
Septiembre-Octubre	$24-30$	9	$216-270$
Noviembre	$40-44$	4	$160-176$
Total		52	$571-719$

Tabla 2.1: Estimación de horas.

A la hora de realizar un presupuesto del proyecto se tendrán en cuenta todos los factores comentados anteriormente. El principal factor a tener en cuenta a la hora de elaborar el mismo son las horas estipuladas por el personal que trabaja en el proyecto. El computador utilizado o los recursos energéticos son factores que también se tendrán en cuenta aunque no se reflejarán en este.

Para reflejar el dinero por hora que cobrará cada rol hemos estipulado que tanto el analista como el diseñador son un perfil junior. Por lo tanto ambos perfiles hemos estipulado que cobran un sueldo de 24,000 € al año brutos, lo que supone 2000 € mensuales entre 160 horas trabajadas, hacen un total de $12,50$ € la hora. Para el Jefe de proyecto hemos estipulado que cobra aproximadamente 40000 € el mes, lo que supone por la misma regla reflejada anteriormente $20,83$ €. Redondearemos ambas cifras a 21 € y 13 € respectivamente por los factores externos que explicamos. El presupuesto se reflejará con IVA incluido directamente, ya que los sueldos considerados se encuentran en bruto. Así mismo, las horas trabajadas están entre $519-719$, pero en el presupuesto reflejaremos 639, un 60 % sobre las 200 horas de diferencia de la aproximación que existe. Esto será una manera de prevenirnos de alguna manera ante posibles imprevistos. El presupuesto que se presentaría al cliente para la realización de este proyecto es de 8689 € IVA incluido.

Rol	Horas	Precio/Hora	Precio
Análisis-Desarrollo	565	13 €	7345 €
Jefe proyecto	64	21 €	1344 €
			8689 €

Tabla 2.2: Presupuesto del proyecto.

Capítulo 3

Conceptos básicos

Para el entendimiento de puntos posteriores se explicará brevemente en este capítulo una serie de conceptos, que se considera conveniente tenerlos lo suficientemente presentes. Esta sección nos permitirá tratar posteriormente temas de gran peso en nuestro documento como la seguridad o el proceso de un protocolo específico.

Definición 3.1. Un problema es una descripción general de una tarea que depende de unos parámetros específicos. Primero se dará contexto al problema indicando los parámetros necesarios, para posteriormente realizar una pregunta de la que se espera una respuesta o solución.

Definición 3.2. Una instancia de un problema se define como un caso particular de un problema al que se le han asignado valores a los parámetros.

Definición 3.3. Un algoritmo es una lista de secuencias que nos servirán para la resolución del problema. Las distintas entradas que tiene el algoritmo son las diferentes instrucciones del problema. El algoritmo debe tener una salida que es la solución al problema.

Definición 3.4. Se dice que un problema es de decisión si el conjunto formado por las posibles soluciones del problema es $B = \{\text{Verdadero, Falso}\}$.

Siempre se buscará que el algoritmo resuelva lo más pronto posible el problema, esto se traduce en que queremos que el algoritmo sea de tiempo polinomial.

Definición 3.5. Un algoritmo se resuelve en un tiempo polinomial si existe $d \in \mathbb{N}$ tal que el tiempo de resolución del algoritmo para una especificación de longitud k es $\mathcal{O}(k^d)$, donde el orden de operaciones del algoritmo es k^d salvo una constante multiplicativa.

3.1. Conceptos Matemáticos

En este capítulo vamos a exponer una serie de teorías y conceptos matemáticos, se utilizará pseudocódigo para explicar de manera sencilla los algoritmos que a continuación se plantean.

3.1.1. Aritmética Modular

Entendemos la aritmética modular, de manera general, como la división de dos números enteros que tienen como interés conocer un cierto valor al que llamaremos módulo.

Definición 3.6. Para realizar la operación modular necesitamos $a, r, m \in \mathbb{Z}$, donde llamamos módulo a $m > 0$ y r al resto resultante de la operación

$$a = r \mod m.$$

Si queremos calcular $\mathbf{a^n} \mod \mathbf{m}$ para valores grandes de n, el cálculo de la exponenciación modular se dificulta bastante. Desafortunadamente, a^n se vuelve muy grande incluso para valores pequeños de n.

¿Cómo se puede calcular $\mathbf{a^n} \mod \mathbf{m}$ de forma rápida para cualquier n?
La idea general de la exponenciación modular rápida reside en calcular el exponente n en dígitos binarios $(d_t, d_{t-1}, \ldots, d_2, d_1)$, con $d_t = 1$, y obtener los distintos cuadrados sucesivos $\mod m$ de la base a: $(a^1, a^2, a^4, \ldots, a^{2t})$, para después multiplicar módulo m las potencias a^{2i} correspondientes a los dígitos binarios d_i que sean "1".
Partiendo de esta idea, en el algoritmo que mostraremos a continuación se calculan y multiplican los cuadrados correspondientes en cada iteración del bucle, a la vez que se va dividiendo el exponente n entre 2 (para hallar los dígitos del exponente n en binario).

Algoritmo (Función Exponenciación)
Entrada: números enteros a, n (exponente), m (módulo), con $0 \leq a < m,\ \ m \geq 2,\ \ n \geq 0$ **Salida**: un entero $exp = a^n \mod m$.

begin
 	
exp:=1
$x := a \mod m$
 	
do{
 	
if "n es impar"$\{exp := (exp * x) \mod m\}$
 	
$x := (x * x) \mod m$
 	
$n := n/2$
 	
} **while** $(n > 0)$
 	
return $\{exp\}$

Ejemplo 3.1. Calcularemos $7^{219} \mod 13$ con el algoritmo anterior de manera sencilla y rápida:

Primero escribiremos $n = 219$ en binario:

$$219 = 11011011$$

Ahora obtendremos los distintos cuadrados sucesivos $\mod m$ de la base a.

$7^1 \mod 13 = 7$

$7^2 \mod 13 = 10$

$7^8 \mod 13 = (7^4 * 7^4) \mod 13$

$7^8 \mod 13 = (7^2 \mod 13 * 7^2 \mod 13 * 7^2 \mod 13 * 7^2 \mod 13) \mod 13$

$7^8 \mod 13 = (10^4) \mod 13 = 3$

$7^{16} \mod 13 = (7^8 * 7^8) \mod 13 = 9$

$$7^{64} \text{ mód } 13 = (7^{16} \text{ mód } 13 * 7^{16} \text{ mód } 13 * 7^{16} \text{ mód } 13 * 7^{16} \text{ mód } 13) \text{ mód } 13$$

$$7^{64} \text{ mód } 13 = (9^4) \text{ mód } 13 = 9$$

$$7^{128} \text{ mód } 13 = (7^{64} * 7^{64}) \text{ mód } 13 = 3$$

Por último multiplicamos los resultados anteriores:

$$7^{219} \text{ mód } 13 = (7^1 * 7^2 * 7^8 * 7^{16} * 7^{64} * 7^{128}) \text{ mód } 13$$

$$7^{219} \text{ mód } 13 = (7 * 10 * 3 * 9 * 9 * 3) \text{ mód } 13$$

$$7^{219} \text{ mód } 13 = (51030) \text{ mód } 13 = 5$$

Un tema de gran relevancia en este punto es calcular el inverso modular, para ello utilizaremos el Algoritmo de Euclides Extendido (EEA).

Este algoritmo quiere calcular $a \cdot x \mod q = 1$, donde $a, x, q \in \mathbb{Z}$ y x es el inverso modular de a. La propiedad primordial que se tiene que cumplir para que se pueda calcular el inverso modular de un número es la siguiente: mcd $(q, a) = 1$.

Para la explicación del Algoritmo de Euclides Extendido necesitamos primero conocer el Algoritmo de Euclides (EA). El Algoritmo de Euclides tiene como idea general, que al dividir dos números enteros a y n, obtenemos un cociente q y un resto r de forma que el mcd (a, n) es el mismo que el mcd (n, r).

Algoritmo (EEA)
Entrada: Los números enteros r_0 y r_1 con $r_0 > r_1 > 0$. **Salida**: El mcd(r_0, r_1), así como s y t tales que mcd $(r_0, r_1) = s \cdot r_0 + t \cdot r_1$. **Parámetros**: $s_0 = 1$, $t_0 = 0$, $s_1 = 0, t_1 = 1$ con $i = 1$.

begin
 	
do{
 	
$y_o = y_o + 1$
 	
$r_i = r_i - 2$ mód $r_i - 1$
 	
$q_i - 1 = \frac{(r_i - 2 - r_i)}{r_i - 1}$
 	
$s_i = s_i - 2 - q_i - 1 \cdot s_i - 1$
 	
$t_i = t_i - 2 - q_i - 1 \cdot t_i - 1$
 	
} **while** $r_i \neq 0$
 	
return {
 	
mcd $(r_0, r_1) = r_i - 1$
 	
$s = s_i - 1$
 	
$t = t_i - 1$
{

Ejemplo 3.2. Empezaremos calculando el algoritmo EA con dos números enteros grandes, tal que $r_0 = 911$ y $r_1 = 301$. Entonces:

$$911 = 3 \cdot 301 + 8 \rightarrow mcd\ (911, 301) = mcd\ (301, 8)$$

$$301 = 37 \cdot 8 + 5 \rightarrow mcd\ (301, 8) = mcd\ (8, 5)$$

$$8 = 1 \cdot 5 + 3 \rightarrow mcd\ (8, 5) = mcd\ (5, 3)$$

$5 = 1 \cdot 3 + 2 \rightarrow mcd\ \ (5,3) = mcd\ \ (3,2)$

$3 = 1 \cdot 2 + 1 \rightarrow mcd\ \ (3,2) = mcd\ \ (2,1)$

$2 = 1 \cdot 1 + 1 \rightarrow mcd(2,1) = mcd(1,1)$

$1 = 1 \cdot 1 + 0 \rightarrow mcd(1,1) = mcd(1,0) = 1$

Como resultado obtenemos que el $mcd\ \ (911, 301) = 1$, entonces se podrá calcular el inverso con el Algoritmo de Euclides Extendido.

Para calcular el inverso de 301 módulo 911 se usarán los mismos datos del ejemplo anterior $r_0 = 911$ y $r_1 = 301$. Entonces:

- $i = 2 \rightarrow 911 = 3 \cdot 301 + 8$
  $r_2 = 8 = [1]911 + [-3] \cdot 301$
- $i = 3 \rightarrow 301 = 37 \cdot 8 + 5$
  $r_3 = 5 = 301 - 37 \cdot 8$
  $r_3 = 301 - 37 \cdot (911 - 3 \cdot 301)$
  $r_3 = -[37]911 + [112]301$
- $i = 4 \rightarrow 8 = 1 \cdot 5 + 3$
  $r_4 = 3 = 8 - 5$
  $r_4 = 1 \cdot 911 + (-3) \cdot 301 - (-37 \cdot 911 + 112 \cdot 301)$
  $r_4 = [38]911 - [115]301$
- $i = 5 \rightarrow 5 = 1 \cdot 3 + 2$
  $r_5 = 2 = 5 - 3$
  $r_5 = -37 \cdot 911 + 112 \cdot 301 - (38 \cdot 911 - 115 \cdot 301)$
  $r_5 = -[75]911 - [227]301$
- $i = 6 \rightarrow 3 = 1 \cdot 2 + 1$
  $r_6 = 1 = 3 - 2$
  $r_6 = 38 \cdot 911 - 115 \cdot 301 - (-75 \cdot 911 + 227 \cdot 301)$
  $r_6 = [113]911 - [342]301$
- $i = 7 \rightarrow 2 = 1 \cdot 1 + 1$
  $r_7 = 1 = 2 - 1$
  $r_7 = -75 \cdot 911 + 227 \cdot 301 - (113 \cdot 911 - 342 \cdot 301)$
  $r_7 = -[188]911 - [569]301$

De esta manera la inversa de 301 módulo 911 es 569 ya que $1 = -188 \cdot 911 + 569 \cdot 301$.

Definición 3.7. Dado dos números enteros a y b, decimos que son coprimos si existe un número entero y tal que $b \cdot y = 1 \mod a$, es decir, b tiene un inverso para el producto modulo a.

Teorema 3.8. *Teorema de Fermat: Sea q un número primo y b un entero tal que q y b son coprimos, entonces*

$$b^{q-1} = 1 \ mod(q).$$

3.1.2. Teoría de grupos

En la computación, el hecho de trabajar con conjuntos infinitos aumenta mucho la complejidad, llegando a ser imposible su uso. Por este motivo los ordenadores trabajan con números enteros, pero con la condición de que un subconjunto de enteros pueda formar un grupo finito. Este grupo nos permitirá mantener propiedades específicas que nos son de gran utilidad.

Para ello definimos formalmente grupo.

Definición 3.9. Llamamos grupo a un conjunto $\mathbb{G}$ junto con una operación binaria $\circ$ que cumple las siguientes propiedades:

- Propiedad asociativa : Para todo $g_1, g_2, g_3 \in G, (g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.
- Existencia de elemento neutro: Existe una identidad $e \in \mathbb{G}$ tal que para todo $g \in \mathbb{G}, e \circ g = g = g \circ e$.
- Cada elemento tiene inverso o simétrico : Para todo $s \in \mathbb{G}$ existe un elemento $h \in \mathbb{G}$ tal que $g \circ h = e = h \circ g$ siendo e el elemento neutro.

Si la operación del grupo se denota mediante "$\cdot$", entonces el grupo, conjunto con operación, se denota por $(G, \cdot)$.

Definición 3.10. Un grupo $\mathbb{G}$ con operación $\circ$ es abeliano o conmutativo si se cumple lo siguiente:

- Propiedad conmutativa : Para todo $g, h \in G, g \circ h = h \circ g$.

Nota 3.11. Si la operación del grupo se denota mediante "$+$", entonces se supone siempre que el grupo $(G, +)$ es conmutativo.

Ejemplo 3.3. Si tomamos $GL(2,\mathbb{R})$ el conjunto de todas las matrices inversibles de orden $2*2$ con coeficientes reales (recordamos que una matriz es inversible si y solo si su determinante es no nulo). Consideramos el producto usual de matrices, entonces $GL(2,\mathbb{R})$ es un grupo, pero no sería un grupo abeliano.

Definición 3.12. Sea $(G,\cdot)$ un grupo, un subgrupo es un subconjunto no vacío $H \subseteq G$ que a su vez es grupo con la operación "$\cdot$" restringida a H.

Lema 3.13. *Sea $(G,\cdot)$ un grupo, un subconjunto $H \subseteq G$ es un subgrupo sí y solo sí se cumplen las condiciones siguientes:*

- *El elemento neutro $e \in H$.*
- *Para todo $x, y \in H$ se cumple que $xy \in H$.*
- *Para todo $x \in H$, se cumple $x^{-1} \in H$.*

Nota 3.14. Todo grupo G tiene, al menos, dos subgrupos: el propio grupo G y el subgrupo constituido solo por el propio elemento neutro $\{e\}$.

Definición 3.15. Un subgrupo N de un grupo G es un subgrupo normal si se cumple que $g^{-1}Ng = N$ para todo $g \in G$. Se denota $N \triangleleft G$. También se llama subgrupo invariante.

Nota 3.16. La definición no indica que $g^{-1}yg = y$ para $g \in G, y \in N$.

Teorema 3.17. *Sea N un subgrupo normal de G; entonces el conjunto cociente G/N es un grupo con la operación $(xN)(yN) = (xy)N$. Se denomina grupo cociente de G entre N.*

Definición 3.18. Para un elemento $g \in G$ el mínimo número $n = 1,2,3,\ldots$ tal que $g^n = 1$ se denomina orden de g y se denota por **ord g**. Si $g^n \neq 1$ para ningún n, se dice que g tiene un orden infinito.

Corolario 3.19. *Si g es un elemento de orden finito, entonces:*

$$ord\ \ g^k = \frac{ord(g)}{mcd(ord(g),k)}.$$

Demostración. Sea $n = ord(g)$. Si $mcd(k,n) = d$, entonces tenemos que:

$n = n'd, k = k'd$, donde $mcd\ \ (n',k') = 1$.
Luego:
$n|km \Leftrightarrow n'd|k'dm \Leftrightarrow n'|k'm \Leftrightarrow n'|m$,
Entonces sustituyendo tenemos: $ord\ \ (g^k) = \text{mín}\ \{m|(g^k)^m = 1\} = \text{mín}\ \{m|n|km\} = \text{mín}\ \{m|n'|m = n\}' = n/d$. □

Nota 3.20. Cuando $\mathbb{G}$ tiene un número finito de elementos, decimos que $\mathbb{G}$ es un grupo finito y $|\mathbb{G}|$ denota el orden del grupo; es decir, el número de elementos en $\mathbb{G}$.

Nota 3.21. Definiremos un subconjunto de números enteros como $\mathbb{G} = \mathbb{Z}_q$ donde q es el modulo entero que marcará el número de elementos del grupo. Las operaciones que se realizan en el conjunto siempre dan como resultado un elemento del mismo.

Definición 3.22. Sean $(G, *)$ y $(K, \circ)$ dos grupos. Una aplicación $f : G \longrightarrow K$ es un homomorfismo de grupos si cumple:

$$f(x * y) = f(x) \circ f(y), \ \ para \ \ todo \ \ x, y \in G.$$

Consecuencias de la definición:

- $f(1_G) = 1_K$
- $f(x^{-1}) = f(x)^{-1}$
- $f(x^n) = f(x)^n$ para todo $n \in \mathbb{Z}$

Definición 3.23. Un isomorfismo es un homomorfismo biyectivo tal que su inversa es también un homomorfismo.

Nota 3.24. Dos grupos, G y K son isomorfos $G \cong K$ si existe un isomorfismo entre ambos grupos. La aplicación inversa de un isomorfismo es otro isomorfismo

Importante: Sea $f : G \overrightarrow{\cong} K$ un isomorfismo de grupos. Entonces G y K tienen exactamente la misma estructura como grupos. Técnicamente, son el mismo grupo, donde, en cada caso, denotamos los elementos de forma distinta. Se cumple lo siguiente:

- Para cada $g \in G$, ord $(g) =$ ord $(f(g))$.
- G y K tienen exactamente el mismo número de elementos de cada orden.
- G y K tienen exactamente el mismo número de subgrupos de cada orden.
- Si $H \subset G$ es un subgrupo, entonces es isomorfo al subgrupo $f(H) \subset K$.

Ejemplo 3.4. Veamos dos isomorfismos claros: Sea $e \in G$ el elemento neutro de un grupo. Entonces $G/\{e\} \cong G$, $G/G \cong \{e\}$.

Llamamos grupo cíclico al grupo que contiene al menos un elemento generador que se encarga de construir todos los elementos del grupo.

Definición 3.25. Se dice que un grupo G es cíclico si existe un elemento $g \in G$ que genera todo G, es decir $G = \langle g \rangle$.

Nota 3.26. Un grupo finito es cíclico si y solo si este posee un elemento de orden $n = \mid G \mid$. En este caso los elementos que componen G son $\{1, g, g^2, \ldots, g^{n-1}\}$.

Proposición 3.27. *Sea $G = \langle g \rangle$ un grupo cíclico finito de orden n. Entonces, otro elemento $g^k \in G$ es un generador de G si y solamente si $mcd\ (k, n) = 1$.*

Demostración. g^k será un generador de G si y solo si $ord\ (g^k) = n$. Por el corolario 3.19:

$$ord\ (g^k) = \frac{n}{mcd\ (k, n)}$$

□

Ejemplo 3.5. Consideramos el grupo $(\mathbb{Z}/n\mathbb{Z}, +)$, es un grupo cíclico ya que todos los elementos se pueden generar a partir de $[1]_n$

$$[0]_n = 0 \cdot [1]_n,$$

$$[2]_n = 2 \cdot [1]_n = [1]_n + [1]_n,$$

$$[7]_n = 7 \cdot [1]_n = [1]_n + [1]_n + [1]_n + [1]_n + [1]_n + [1]_n + [1]_n,$$

En general $[k]_n$ es un generador del grupo $\mathbb{Z}/n\mathbb{Z}$ si el $mcd\ (k, n) = 1$

Ejemplo 3.6. El grupo $(\mathbb{Z}, +)$, es cíclico ya que todo número entero puede ser generado por los elementos $+1$ y -1.

Nota 3.28. En general, si G es un grupo cíclico infinito, los únicos generadores serán g y g^{-1}.

Los ejemplos que hemos mostrado hasta el momento son todos los grupos cíclicos posibles, salvo isomorfismo. Por eso es tan importante la siguiente proposición:

Proposición 3.29. *Todo grupo cíclico finito de orden n es isomorfo a $\mathbb{Z}/n\mathbb{Z}$.*

Demostración. Sea G es un grupo cíclico finito de orden n, entonces $G = \langle g \rangle = \{1, g, g^2, \ldots, g^{n-1}\}$ para algún $g \in G$. Definamos la aplicación:

$$f : G \longrightarrow \mathbb{Z}/n\mathbb{Z}$$

$$g^k \longrightarrow [k]_n.$$

Esta aplicación f es una biyección dado que está bien definida: $g^k = g^\ell$ si y solo si $k = \ell(mod\ \ n)$. Además, sería un homomorfismo:

$$f(g^k \cdot g^\ell) = f(g^{k+\ell}) = [k+\ell]_n = [k]_n + [\ell]_n = f(g^k) + f(g^\ell).$$

□

Proposición 3.30. *Todo grupo cíclico infinito es isomorfo a* $(\mathbb{Z}, +)$.

Demostración. Sea G es un grupo cíclico infinito, entonces $G = \langle g \rangle = \{g^n | n \in Z\}$, y la aplicación:

$$G \longrightarrow \mathbb{Z}$$
$$g^n \longmapsto n$$

es un isomorfismo.

□

Ejemplo 3.7. El grupo alternado $A_3 = \{id, (123), (132)\}$, es cíclico, ya que se puede generar a partir del (123) o (132). Este grupo sería isomorfo a $(\mathbb{Z}/3\mathbb{Z}, +)$.

Definición 3.31. Un anillo conmutativo con unidad es un conjunto A dotado de dos operaciones binarias internas (suma y producto), que denotamos $(A, +, \cdot)$, y que cumple las siguientes propiedades:

- $(A, +)$ con el elemento neutro 0 es un grupo abeliano.
- Propiedad asociativa del producto: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ para todo $a, b, c \in A$.
- Existe el elemento neutro 1 para el producto: $1 \cdot a = a \cdot 1 = a$ para cada $a \in A$.
- Cuando mezclamos las dos operaciones de grupo, se cumple la propiedad distributiva:
  $a \cdot (c + d) = a \cdot c + a \cdot d, (h + d) \cdot a = h \cdot a + d \cdot a$ para todo $a, c, d, h \in A$.
- El producto es conmutativo: $a \cdot b = b \cdot a$ para todo $a, b \in A$.
  (Si está propiedad no se cumple diríamos que el anillo no es abeliano).

Ejemplo 3.8. $\mathbb{Z}, \mathbb{Z}/(n)$ son anillos con su suma y producto usuales.

Un cuerpo es una estructura algebraica que forma un grupo aditivo y un grupo multiplicativo, cada uno con su operación respectiva de grupo: suma y multiplicación. En un cuerpo también podemos crear operaciones como la división y la resta; la resta se puede formular mediante la suma de un número negativo y la división se puede formular utilizando la inversa.

Definición 3.32. Un anillo A se denomina cuerpo si cada elemento no nulo $a \in A \setminus \{0\}$ tiene inverso en A. Es decir, existe $c \in A$ tal que $a \cdot c = c \cdot a = 1$.

Definición 3.33. Un cuerpo finito es un cuerpo A que contiene un número finito de elementos $\mid A \mid < \infty$.

Todo cuerpo finito es conmutativo. La propiedad más importante de los cuerpos finitos es que las operaciones que se realizan dentro del cuerpo, tienen como resultado elementos del mismo. El número de elementos de A se denomina **cardinalidad de A**.

3.2. Herramientas criptográficas

Ahora hablaremos de una serie de conceptos criptográficos que nos servirán para explicar la codificación y la seguridad de las siguientes secciones.

Se empezará definiendo un concepto perteneciente a la teoría computacional, los sistemas de pruebas interactivas. Este tipo de pruebas están formadas por dos participantes no únicos a los que llamaremos *probador* P y *verificador* V. Estos intercambian un número finito de información siempre con el objetivo de que P le demuestre a V una instancia de un cierto problema de decisión es Verdadera. En este tipo de pruebas produce continuamente una serie de preguntas del conocimiento que tiene P. V suele ser una máquina que computacionalmente está limitada.

En todo protocolo puede haber participantes deshonestos que intenten romper el proceso. Por eso nos podemos encontrar ante un probador o un verificador deshonesto. Una vez finalizado todo el proceso de intercambio de mensajes entre el verificador y el probador, V tiene que decidir si la instancia es Verdadera o es rechazada por ser Falsa. Este tipo de pruebas tienen como dificultad que a la hora de entrar en juego un tercer participante que también quiere saber el secreto del verificador, este puede pensar que P y V están asociados. Por esto el tercer participante en cuestión necesita que el Probador le demuestre sus conocimientos realizando la misma prueba interactiva.

Esto quiere decir que ningún participante diferente a P y V podrá asegurar que estos no están asociados. Si no se produce esta interacción entre P y V, este tipo de pruebas se llaman no interactivas. Se evitará el principal problema de las pruebas interactivas, dado que si actúa un tercer participante no pensará que puede haber un asociamiento entre P y V. En este tipo de pruebas se suelen utilizar maquinas y programas adicionales que marcarán el orden de los experimentos.

3.2.1. Prueba de conocimiento cero

Empezaremos explicando la prueba de conocimiento cero. Este protocolo consta de un participante al que llamaremos probador P, el cual es capaz de probar que cierta declaración es verdadera a otro participante que llamaremos verificador V. A la hora de realizar esta acción el probador utiliza conocimientos adicionales y el verificador se encarga de aceptar o rechazar la prueba. Todo este proceso se realiza sin revelar ningún tipo de información.

La prueba de conocimiento cero debe satisfacer una serie de propiedades:

- **Completitud**: Si la explicación es verdadera y el verificador es un participante honesto que no trata de incomodar el proceso, aceptará dicha aclaración.
- **Solidez**: Si la aclaración es errónea, debería fallar con gran probabilidad.
- **Conocimientos cero**: Si la exposición es verdadera, los verificadores no podrán aprender ninguna información más allá que la declaración es verdadera.

Se expone un ejemplo intuitivo para entender mejor este protocolo criptográfico.

Ejemplo 3.9. Consideremos que dos amigos, Ana y Bob, quedan en una cueva, la cual tiene un camino principal que se divide en dos pasillos, y al final de cada pasillo estos se vuelven a unir, teniendo la cueva una forma de anillo. En la unión entre los dos pasillo se encuentra una puerta que solo se abre con una palabra mágica, por lo tanto, solo se podrá pasar del pasillo A al pasillo B si se tiene conocimiento de la palabra mágica.

Ana le afirma a Bob que sabe la palabra secreta que abre la puerta y que se la revelará si le da unos cromos a cambio de esta información. Bob no se fía de Ana y dice que antes de darle los cromos tiene que decirle la palabra secreta. Ana no piensa revelar la palabra mágica antes de cobrar su recompensa y llegan a un acuerdo: Ana entrará a la cueva, cogerá uno de los pasillos el A o el B sin decir cual eligió a Bob, y segundos más tarde entrará Bob para intentar verificar que Ana no le miente y conoce la palabra secreta. Una vez dentro Bob le pedirá a Ana que regrese al principio de la cueva por el pasillo A o el B. Si Ana conoce realmente la palabra que abre la puerta siempre podrá regresar por el pasillo que le indique Bob, si es necesario abriendo la puerta que comunica un pasillo con otro. En caso de no conocer la palabra, tendrá 50 % de posibilidades de adivinar el pasillo por el que solicitará Bob que regresé al principio de la cueva. Como Ana podría haber tenido suerte realizarán

25 repeticiones, por lo que Ana disminuye su probabilidad de acertar todas las veces escasamente a $(1 \setminus 2)^{25}$.

Después de realizar todo el proceso Ana siempre consigue volver por el camino indicado por Bob. Eva que era amiga de Ana y Bob lleva todo el rato espiando a estos. Eva tiene la duda de si Ana sabe la palabra secreta, o esta aliada con Bob. Por eso sale de su escondite para hablar con Bob y preguntarle si conoce la palabra secreta, y que le dará unas gominolas a cambio, si este se la otorga. Como Ana todavía no reveló a Bob la palabra secreta, no puede repetir la prueba a Eva, y solo podrá demostrárselo Ana. Esto implica que todo participante distinto al probador y verificador no estará seguro de que no exista una conspiración en la prueba. Entonces, se tendrá que repetir todo el proceso entre Ana y Eva para que este participante se sienta seguro de que no existe tal asociación entre Ana y Bob.

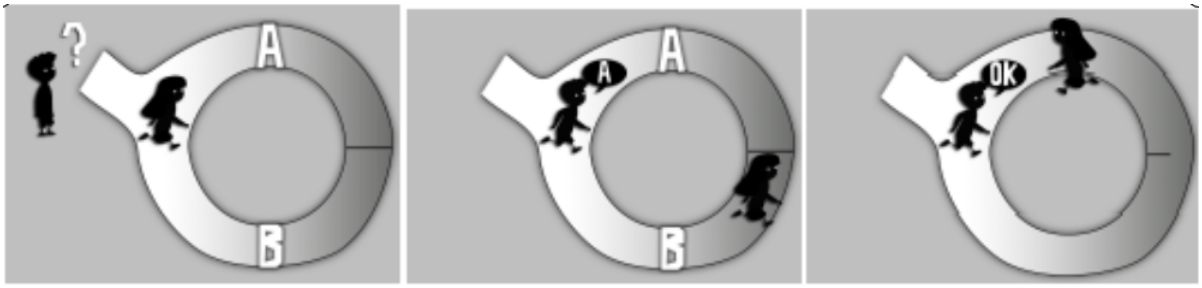

Imagen 3.1: Prueba de conocimiento Cero.

Nota 3.34. Este ejemplo que hemos dado sería una prueba de conocimiento cero interactiva. Para que el ejemplo anterior fuera una prueba de conocimiento cero no interactiva se eliminará a Bob y en su lugar añadiremos dos puertas más en la cueva, cada una situada en la entrada de cada pasillo. Las tres puertas que ahora existen en la cueva se abrirán todas con la misma palabra clave que supuestamente tiene en conocimiento Ana.

Ana entrará en la cueva, tomará el pasillo A o B, y cuando llegue como en el caso anterior a la puerta que une el pasillo A con el B, se encontrará con una tablet al lado de la puerta que le dirá porque lugar tiene que regresar, además de la tablet indicar el orden del experimento, independientemente se cerrará una de las dos puertas al azar de la entrada de los dos pasillos. Esto supondría que la posibilidad de que Ana no sepa la palabra clave y pueda salir de la cueva es $1 \setminus 4$. La única posibilidad que

puede existir para que Ana entre en la cueva y pueda salir sin conocimiento de la palabra mágica, es la siguiente: si Ana coincide en la elección del pasillo de entrada y el pasillo de regreso que marque la tablet, y a la vez tenga la suerte de que la puerta que se cierre al azar sea distinta al pasillo por el que le mandó regresar la tablet. Si se dan estas casualidades Ana sí podría salir de la cueva sin el conocimiento de la palabra que abre las puertas, porque casualmente no necesitó la apertura de ninguna de ellas para salir de la cueva.

De esta manera se evitará la interacción entre Bob y Ana. Este proceso podemos hacer que se repita como en el caso anterior unas 25 repeticiones para que la prueba sea efectiva. El proceso de verificación se mantendría totalmente en secreto, y Ana tiene que seguir demostrando que tiene plenos conocimientos sobre la palabra mágica. Este secretismo sobre la verificación del secreto permite que Eva nunca pueda pensar como antes que Ana y Bob están pactando todo para engañarla a ella.

Uno de los problemas más representativos de las Pruebas de Conocimiento Cero es el Problema del logaritmo discreto que mostraremos a continuación.

3.2.2. Problema del logaritmo discreto

Siempre que tengamos una función f y un valor x nos será muy sencillo calcular $f(x)$. Los problemas surgen cuando tenemos $f(x)$ y queremos calcular x, entonces nos sería necesario calcular la inversa de $f(x)$. Calcular la función $f(x)$ puede resultar algo sencillo, pero calcular $f^{-1}(x)$ puede llegar a ser un procedimiento mucho más costoso.

Definición 3.35. Una función $f()$ es una función unidireccional si:

$y = f(x)$ es computacionalmente sencillo de resolver.

$x = f^{-1}(y)$ es computacionalmente costoso de resolver.

Un ejemplo de una función unidireccional es la función logaritmo discreto.

Definición 3.36. Dado un grupo G, un generador g y $c \in G$, se define el Problema de logaritmo discreto (**DL**), como el problema en el que es necesario encontrar un entero a, tal que $g^a = c$.

Daremos un ejemplo numérico muy sencillo de este logaritmo:

Ejemplo 3.10. Dado un grupo $G = \mathbb{Z}_{64}$, un generador $g = 7$, y un elemento $c = 9$, entonces resolvemos:

$$7^a = 9 \mod 64$$

Entonces $a = 7$ ya que tenemos:

$$7^7 = 9 \mod 47$$

¿Pero, qué ocurriría si el grupo es distinto de $\mathbb{Z}_n$?
Para estos casos hay distintos tipos de ataques, nosotros hablaremos de los más comunes en la siguiente sección.

3.2.2.1. Resolución del problema del logaritmo discreto

Sabemos que somos capaces de resolver el problema DL, solo pediremos que esto no se realice dentro del tiempo polinomial, dado que esto hará mas débiles nuestros posteriores protocolos. Podemos tomar como solución aumentar el tamaño de los grupos con los que operamos, esto provocaría un mayor tiempo de ejecución del protocolo.

Ahora veremos diferentes tipos de algoritmos para resolver el problema DL. Todos ellos tienen la misma entrada y salida. Por una parte, la entrada es un grupo cíclico G, un generador g y un elemento $c \in G$, de manera que podemos calcular $t = ord(G)$. La salida es un número entero a que resuelve $g^a = c$.

1. Búsqueda por fuerza bruta

Como su nombre indica el algoritmo agota todas las posibles soluciones hasta que se llega a la correcta, para ello se calcula $\{g^0, g^1, g^2, g^3 \dots g^{t-1}\}$ y se prueba si $g^i \stackrel{?}{=} c$. El principal problema es que el tiempo de ejecución es de $\mathcal{O}(n)$, es decir, el orden de operaciones del algoritmo es n salvo una constante multiplicativa. Esto sería demasiado alto para calcular potencialmente cada única solución.

Ejemplo 3.11. En el ejemplo 3.10 se podían haber probado todas las posibles soluciones $\{g^0, g^1, g^2, \dots, g^{63}\}$ hasta encontrar $7^a = 9 \mod 64$, pero esto sería muy costoso y carecería de sentido, ya que el problema DL se complica en grupos $\mathbb{Z}_p$ con una p lo suficientemente grande.

2. Ataques de raíz cuadrada

Este algoritmo tiene como base dividir todos los problemas de DL en pequeños problemas. Esta forma obtiene mejores resultados a la hora de tratar problemas más sencillos. El tiempo de ejecución es de aproximadamente $\mathcal{O}(\sqrt{n})$. En este grupo

encontramos el algoritmo Baby-steps Giant-steps.

2.1 Algoritmo Baby-steps Giant-steps

Se suele operar en $\mathbb{Z}_q^*$, entonces nos centraremos en este grupo a la hora de explicar los alagoritmos. La idea reside en dividir el grupo en subgrupos más pequeños. Si tenemos un grupo cíclico G, un generador g y un elemento $c \in G$, entonces como idea global podemos imaginarnos los elementos en G como puntos en un círculo de tal manera que $1 = g^0, g^1, g^2, \ldots, g^{n-2}, g^{n-1}, g^n = 1$.

Si dividimos el círculo en intervalos $u = [\sqrt{q}]$, entonces los Giant-steps serían los intervalos. Por el contrario, los elementos en los intervalos que como máximo podrían ser u, significaría los Baby-steps. Ahora viendo el exponente $a = ui + j$ donde $0 \leq i, j \leq u$, podemos replantear el problema como:

$$c = g^a = g^{ui+j} = g^{ui} \cdot g^j$$

$$c(g^{-u})^i = g^j.$$

Tenemos como principal objetivo encontrar un j e i tal que $c(g^{-u})^i = g^j$, para ello podemos calcular g^j para cada $j = \{0, 1, \ldots, u-1\}$ y $c(g^{-u})^i = g^j$ para cada $i = \{0, 1, \ldots, u-1\}$. Se intentará buscar la coincidencia entre las dos listas. Esto explicaría una idea general, una visualización más detallada se presenta con el ejemplo que daremos a continuación y con el algoritmo en pseudocódigo presentado al final de la sección.

Ejemplo 3.12. Sea $q = 31, g = 3$ y $c = 5$.

Paso 1: $u = [\sqrt{q}] = 6$

Paso 2: Calculamos $1, g, g^1, \ldots, g^4$ y tenemos $0 \leq j \leq u - 1$ y podemos calcular g^j mód q. Entonces:

$$g^0 \longrightarrow 1, \quad g^1 \longrightarrow 3, \quad g^2 \longrightarrow 9, \quad g^3 \longrightarrow 27, \quad g^4 \longrightarrow 19$$

Paso 3: Calculamos $g^{-u} = 3^{-6}$, podemos usar el algoritmo de Euclides para encontrar $g^{-1} = 3^{-1} = 21$. Con este resultado obtenemos $3^{-6} = 21^6 = 2 \mod 31$.

Paso 4: Usando el resultado del paso anterior podemos calcular eficientemente $c(g^{-u})^i$ para una i creciente hasta que encontremos una coincidencia con el resultado

del **Paso 2**. Sabiendo que $0 \leq yo \leq u-1$ y que $c(g^{-u})^i = 5 \cdot 2^i \mod 31$.

$$5 \cdot 2^0 \longrightarrow 5, \quad 5 \cdot 2^1 \longrightarrow 10, \quad 5 \cdot 2^2 \longrightarrow 20, \quad 5 \cdot 2^3 \longrightarrow 9$$

Encontramos una coincidencia como $c(g^{-u})^3 = g^1 \mod q$.

Paso 5: Entonces podemos calcular $a = g^{iu+j} = g^{3u+1} = g^{3\cdot 6+1} = g^{19} = 3^{19} = 12$ mód 31.

Algoritmo: Baby-Steps y Giant-Steps
Entrada: Grupo G de orden p, un generador g, un elemento $c \in G$. **Salida**: Un entero a tal que $g^a = c$.
begin $u = [\sqrt{q}]$ **for** $j = 0$ to $u-1$ **do** $\{$ Calcula g^j mod q Almacena el par (j, g^j) con g^j como clave en una tabla t $\}$ Calcula g^{-u} mod q **for** $i = 0$ to $u-1$ **do** $\{$ Calcula $x = c(g^{-u})^i$ **if** x esta en la tabla t **then** $\{$ **return** $a = ui + j$ $\}$ $\}$

3.2.3. Diffie-Hellman

Como el cálculo del logaritmo discreto no es sencillo en muchos grupos, utilizaremos el intercambio de claves Diffie-Hellman (**DHP**) para ilustrar el problema del logaritmo discreto. Explicaremos a continuación el algoritmo del intercambio de claves Diffie-Hellman.

Intercambio de claves Diffie-Hellman.

Dado un grupo G, un generador g, y tres participantes Ana, Bob y Eva. Eva ocupará el papel de adversario en el protocolo.
Representaremos en el siguiente esquema general el proceso:

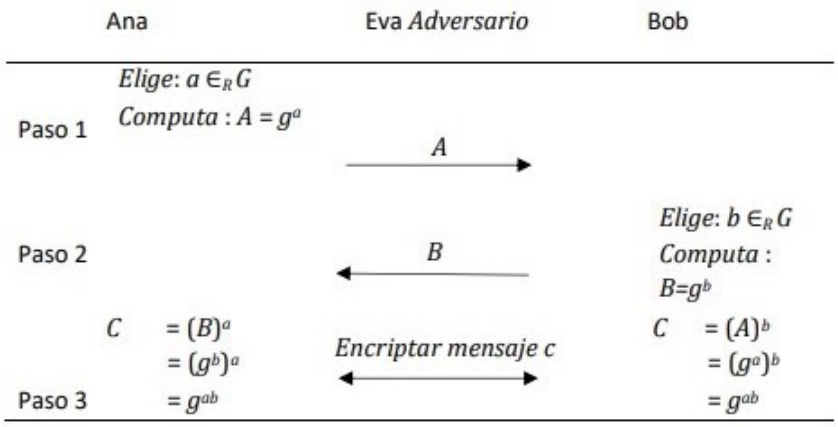

Imagen 3.2: Intercambio de claves Diffie-Hellman.

En el **Paso 1** podemos observar que Ana y Bob son totalmente independientes, entonces eligiendo un elemento aleatorio del grupo G se crea el problema $\mathbf{DL}(\mathbf{A})$ que se envía a Bob.

En el **Paso 2** Bob realiza la misma acción que Ana en el paso 1 y envía el problema $\mathbf{DL}(\mathbf{B})$ a Bob.

En el **Paso 3**, ambos de forma independiente con el problema recibido $\mathbf{DL}(\mathbf{A})$ o $\mathbf{DL}(\mathbf{B})$, se crea C. Tanto Ana como Bob calcularán el mismo valor C que puede ser utilizado como clave de un esquema criptográfico.

Ahora veremos el proceso desde el punto de vista de un personaje independiente llamado Eva que contiene como información G y g, además de A y B. Entonces si Eva

tiene la capacidad de calcular $C = g^{ab}$ podría descifrar cualquier mensaje enviado entre Ana y Bob.

Definición 3.37. Dado un grupo G, y un generador g de tal manera que $A = g^a$ y $B = g^b$. Si seleccionamos $a, b \in \mathbb{Z}_q$ de forma independiente y aleatoria, se define el Problema computacional de Diffie-Hellman (**CDH**), como el problema en encontrar $C = g^{ab}$.

Nota 3.38. Si Eva conoce un algoritmo eficiente para resolver el problema de DL, entonces esta también puede resolver CDH. Si Eva coge el valor a de $A = g^a$ y el b de $B = g^b$, puede calcular C de la misma manera que Ana y Bob. Este hecho implica el siguiente lema.

Lema 3.39. *El problema de CDH no es más complicado que el problema de DL ya que resolviendo el segundo podemos resolver el problema computacional de Diffie-Hellman, pero en general al revés no es cierto, aunque a veces ambos problemas pueden llegar a ser equivalentes.*

Nota 3.40. El problema CDH no garantiza que la solución sea correcta para un elemento de grupo y la resolución de una instancia de CDH. Si tomamos un g^a, g^b y g^c, se seguirá cumpliendo que $c = ab \mod q$. Definimos esto como el problema decisional Diffie-Hellman (**DDH**).

Definición 3.41. Dado un grupo G, un generador g, definimos el Problema decisional de Diffie-Hellman (DDH), de tal manera que eligiendo aleatoriamente y de forma independiente a y b de $\mathbb{Z}_q$, tenemos que $A = g^a, B = g^b, C = g^c$, donde $c = ab$ nos preguntaremos si c es uniformemente aleatorio de $\mathbb{Z}_q$.

Entonces si Eva calcula un valor C y se lo presenta a Ana, Ana puede ver si el valor C es válido, ya que puede calcular $C = g^{ab}$ y podría compararlo con el valor que le fue otorgado por Eva.

Por último Ana responde a Eva con el valor de C y Eva solo podría verificar que esto es cierto resolviendo el problema CDH. Esto nos hace llegar a la conclusión de que el problema CDH no es más sencillo que el problema DDH.

3.2.4. Funciones Hash

Las funciones Hash son un tipo de funciones que reciben como entrada un mensaje de tamaño arbitrario, y tras realizar una serie de operaciones sobre el mensaje, representan como salida una cantidad fija de bits. Estos bits son muy difíciles de descifrar sin ejecutar dicha función, y la salida representa un resumen de toda la información

que recibió como entrada.

Solo se puede volver a crear una salida con esos mismos datos de entrada si se utiliza la misma función hash. Por eso es bastante costoso invertir la salida en la entrada dada, es decir, existen muy pocas posibilidades de calcular el mensaje de origen desde un hash ya formado. Esto es gracias a que el proceso de creación de hashes, un proceso de un solo sentido.

Ejemplo 3.13. Un ejemplo muy sencillo en la vida real para entender esta herramienta criptográfica es la elaboración de una tarta de forma artesana. Los ingredientes que utilicemos para la realización de la tarta, serían el mensaje de entrada. Todo el proceso de cocinado, se asemejaría a la codificación de dichos datos (ingredientes) por la función hash. Al acabar obtenemos como resultado una tarta imposible de repetir sino introducimos los mismo ingredientes en su misma proporción, y el proceso contrario de convertir la tarta en los ingredientes iniciales, viene a ser casi imposible de realizar.

Las funciones hash que utilizaremos son resistente a colisiones cuya definición es la siguiente.

Definición 3.42. Las funciones hash toman un mensaje de tamaño arbitrario y como consecuencia generan un valor de tamaño fijo. Este valor se llama hash, y para el mismo mensaje y función es siempre idéntico. Si se produce un cambio en el mensaje, el valor hash debería de cambiar. Además en estas funciones es improbable calcular dos entradas x y x_0 tal que $H(x) = H(x_0)$ y $x \neq x_0$, por eso el nombre de funciones hash resistentes a colisiones.

Nota 3.43. Una función hash: $\{0,1\}^{\leq L} \longrightarrow \{0,1\}^{\ell}$ se llama resistente a colisiones si es difícil encontrar $x \in \{0,1\}^{\leq L}$ y $x_0 \in \{0,1\}^{\leq L}$ tal que $x \neq x_0$ y $H(x) = H(x_0)$. El valor (x, x_0) se llama colisión. Entonces $\{0,1\}^{\leq L}$ denota el conjunto de cadenas de bits, que presentan como longitud máxima L. Por lo tanto, si $L \geq \ell$ existen colisiones.

Ejemplo 3.14. Ahora mostraremos un ejemplo visual de estas funciones, para ello usaremos dos de las funciones hash más conocidas MD5 y SHA-256:

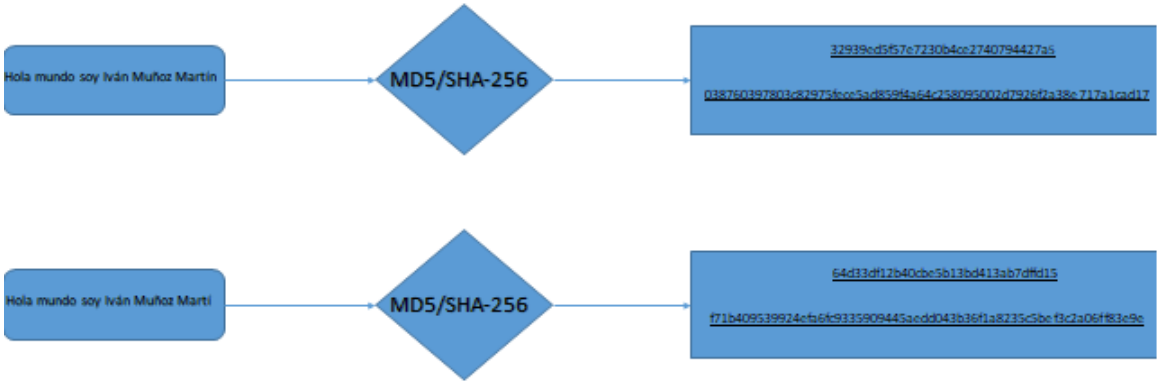

Imagen 3.3: Función Hash.

Nota 3.44. Observando el ejemplo anterior llegamos a la conclusión de cada entrada de datos da como resultado un hash único, para los casos de MD5 y SHA-256. En la segunda entrada solo cambiamos el mensaje eliminando la última letra de la frase, es una modificación mínima, pero se puede observar que se alteró completamente el resultado final de los hashes para MD5 y SHA-256. Esto implica que los hashes son únicos y ningún participante deshonesto podrá realizar hashes de forma sencilla. Estas observaciones nos dan la seguridad de utilizar las funciones hash en aplicaciones complejas.

Las actuales funciones hash tienen un alto nivel de seguridad, aunque esto no significa que sean infalibles. Estas funciones tienen numerosas aplicaciones en la actualidad, desde procesos sencillos como proteger contraseñas, hasta procesos más complejos como la minería y los contratos inteligentes en el nuevo mundo de las criptomonedas y del blockchain.

En Bitcoin la minería hace un uso intensivo de cálculo de hashes SHA-256. Para esto los mineros se encargan de de calcular millones de hashes para crear nuevos bloques Bitcoin. Las funciones Hash también se utilizan en el mundo de las criptomonedas para verificar las transacciones que se hacen en la red.
Las blockchains como Ethereum o Cardano, hacen uso de contratos inteligentes para potenciar distintas aplicaciones. Estos contratos pueden tener datos que no se quieren revelar y que deben mantener su privacidad. El uso de funciones hash en los contratos

inteligentes nos permite que el contrato sea público pero exista información privada dentro de él. Además las funciones hash nos permitirán dar validez y autenticidad al contrato, reconociendo la modificación del mismo.

3.2.5. Fiat-Shamir

Fiat-Shamir es una técnica criptográfica para convertir las pruebas de conocimiento cero interactivas en no interactivas. La información nunca es revelada y nos permite convertir un protocolo de conocimiento cero interactivo entre un probador P y un verificador V en un protocolo no interactivo. El protocolo Fiat-Shamir puede utilizar las funciones hash para proteger la información y su seguridad dependerá de si el protocolo no interactivo que se obtiene como resultado es totalmente sólido.

El protocolo tiene como idea general que el verificador V envía un desafío aleatorio al probador P en cada ronda, y este se encarga de calcular el protocolo original. Por último el probador se encarga de enviar la transcripción del protocolo al verificador, que como su nombre indica se encarga de verificar la validez del protocolo.

Protocolo Fiat-Shamir

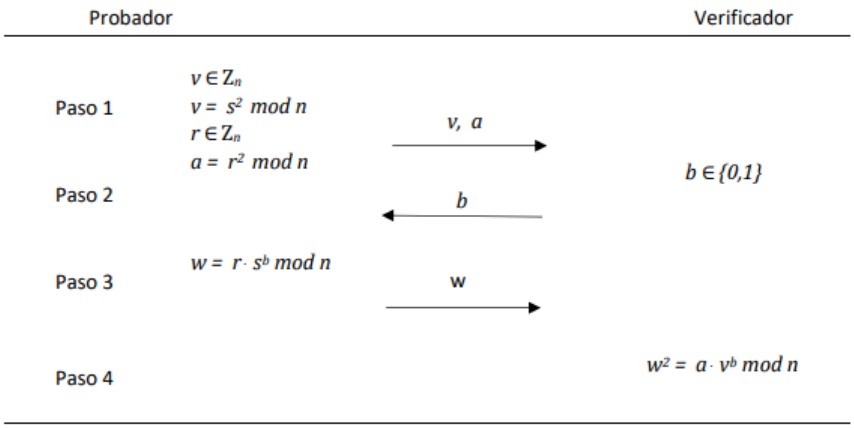

Imagen 3.4: Algoritmo Fiat-Shamir.

El probador P selecciona $n = pq$ con p y q primos grandes aleatorios, n será público mientras que p y q solo se encuentra en conocimiento de P.

En el **Paso 1** P elige aleatoriamente su secreto $s \in \mathbb{Z}_n$, de tal manera que el $mcd\ (s,n) = 1$. En este paso P genera su clave pública $v := s^2 \mod n$, a partir de esta clave se verificará si conoce o no el secreto. Entonces v será público y por el contrario s sera secreto. P deberá convencer a V que conoce s, para ello elige aleatoriamente el compromiso $r \in \mathbb{Z}_n^*$, y calculará la información extra que será el certificado $a = r^2 \mod n$. P envía a y v a V.

En el **Paso 2** V elige aleatoriamente el desafío $b \in \{0,1\}$ y lo envía a P.

En el **Paso 3** P se encarga de calcular la respuesta al desafío $w = rs^b \mod n$, y se la envía a V.

- Si $b = 0$, P envía el valor $w = r$ a V.
- Si $b = 1$, P envía el valor $w = r \cdot s \mod n$ a V.

En el **Paso 4** V se encarga de comprobar si P conoce el secreto s, esto ocurre si $w^2 = av^b \mod n$.

- Si $b = 0$, V comprueba si el valor $a = w^2 \mod n$.
- Si $b = 1$, V comprueba si el valor $a = w^2 \cdot v \mod n$.

El procedimiento se repite t veces en la que los parámetros n, s y v se mantienen invariantes. P tiene que superar las t rondas, para que V de por válida la prueba, si falla solo una vez V rechazará la prueba.

Nota 3.45. Las funciones hash son utilizadas en el protocolo Fiat-Shamir para cifrar el desafio (b) en el **Paso 2**. Se sustituye el desafió b por un resumen utilizando la función hash. Esto permite que el verificador sepa en todo momento que P no esta falseando las pruebas, ya que P no sabe en ningún momento el valor del hash. Un probador deshonesto tiene $1 \setminus 2$ de posibilidades de adivinar en cada ronda si el valor es $b = 0$ o $b = 1$.

Ejemplo 3.15. Vamos a considerar el siguiente ejemplo de prueba de conocimiento cero interactivo y veremos como aplicando Fiat-Shamir el protocolo se convierte en no interactivo.

Tenemos como participantes a Ana, que actúa como probador, esta tratará de demostrar que conoce x tal que $y = g^x$ a Bob, que actúa como verificador.

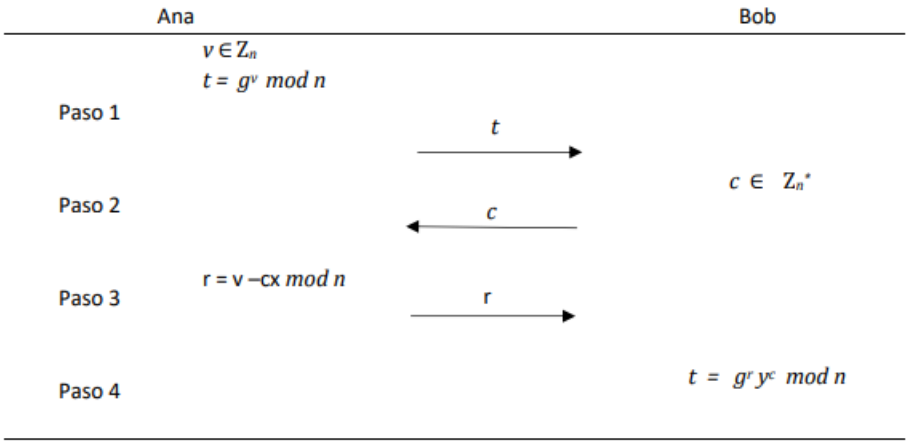

Imagen 3.5: Protocolo interactivo.

Este proceso de prueba anterior es interactivo, dado que en el **Paso 2** Bob envía un número aleatorio c a Ana. Si todos los pasos permanecen invariantes, y sustituimos el Paso 2 por el calculo $c = H(g, y, t)$ realizado directamente por Ana, con $H()$ como función Hash. El ejemplo anterior se convierte en un proceso no interactivo, en el que cualquiera podrá comprobar que $t = g^r y^c$.

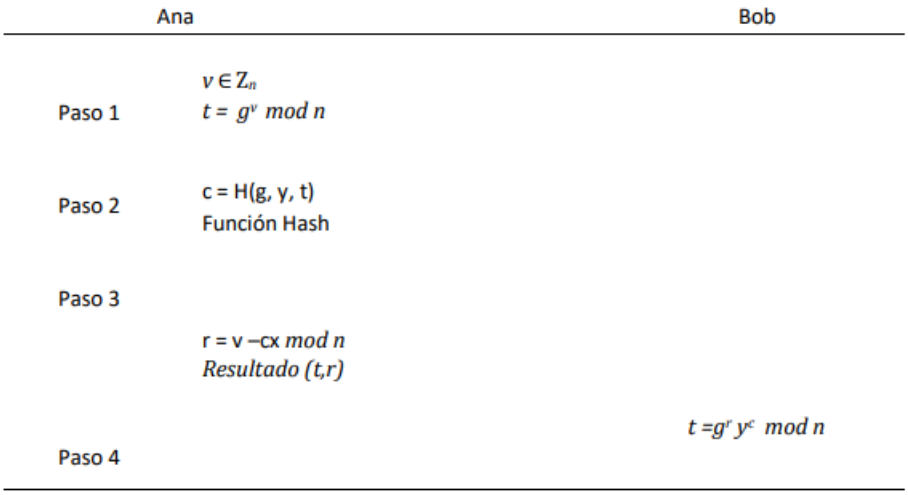

Imagen 3.6: Ejemplo Fiat-Shamir, protocolo no interactivo.

Capítulo 4

Esquemas Criptográficos

4.1. Esquemas de Comparticion de secretos.

4.1.1. Introducción

Los esquemas de compartición de secretos surgen de la importancia de proteger la información frente a observadores no autorizados. De esta manera se hace necesario el uso de métodos robustos para tener acceso a estos archivos. En un esquema de compartición de secretos (ECS) se fragmenta en particiones un dato o "secreto" (como puede ser una clave numérica) entre los participantes de la comunicación de un conjunto, de forma que solo los subconjuntos autorizados pueden reconstruir el secreto a partir de sus particiones. Estas particiones se realizan con el fin de proteger la información. Si logramos juntar un número lo suficientemente grande de particiones, esto nos permitirá reconstruir el secreto por completo. Entonces o bien sabemos todo sobre el secreto o no sabemos nada sobre él.

Existen diversos esquemas de compartición de secretos (ECS), podemos ver la relación que existe entre ellos en la figura 4.1. La línea punteada indica esquemas que son derivados de los ECS más generales [16][31]. Nosotros basaremos nuestro documento en los esquemas de umbral y los esquemas públicamente verificables, aunque haremos una breve explicación de los diferentes (ECS) y sus propiedades.

- **Los esquemas de umbral** son básicamente los introducidos por Shamir (mediante Interpolación de Polinomios) y Blakley (el cual se basa en la Geometría Proyectiva). En los esquemas de umbral el número de particiones es mayor o igual al número de posibles secretos.
- **Los ECS de estructura de acceso general**, solo los miembros de la estructura de acceso (EA) pueden reconstruir el secreto. En concreto, podemos representar a los esquemas de umbral como una EA general.

- **Los ECS verificables** son aquellos en los cuales se prueba la validez de la partición, extenderemos este tipo de esquemas a los esquemas verificables públicamente $PVSS$, en los que cualquier participante podrá verificar las particiones.

- **Los ECS proactivos** son esquemas en los cuales las particiones son actualizadas por los participantes para evitar ataques móviles. La motivación principal se debe a que, en la compartición de secretos, solo se distribuyen una vez los fragmentos del secreto y permanecen fijos en adelante. Los ECS proactivos también se caracterizan por el hecho de que la EA permanece igual antes de y después de la actualización.

- **Los ECS perfectos** son esquemas en los que existe el mismo número de participantes que de posibles secretos. Por ejemplo, los esquemas umbral de Shamir o Blakley son esquemas perfectos.

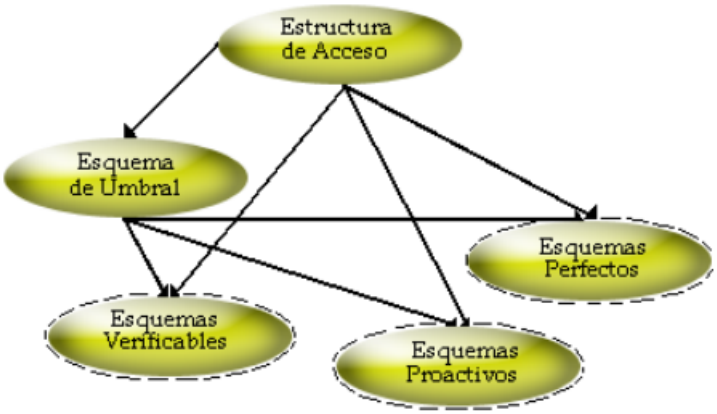

Imagen 4.1: Relación entre diversos ECS

La compartición de secretos tiene una gran relevancia en la criptografía, y en el modelo básico de compartición de secretos podemos distinguir entre dos protocolos a seguir:

(i) **Un protocolo de distribución**, en el que existe un distribuidor D que se encarga de distribuir particiones del secreto S entre el conjunto finito de participantes P, de tal manera que $D \notin P$ y $n(S)$ son las partes en las que se divide el secreto.

(ii) **Un protocolo de reconstrucción**, que permite recuperar el secreto, agrupando las particiones de un subconjunto cualificado de los participantes. Esto lo veremos en la siguiente sección con la explicación de los esquemas de umbral.

Nota 4.1. Las particiones en las que divide D el secreto S, tienen que ser distribuidas de manera confidencial. Ningún participante tiene acceso a la información de otro participante. Esto implica que la posterior reconstrucción del secreto solo se hace posible si contenemos un mínimo de participantes con sus respectivas particiones.

4.1.2. Esquemas de Umbral

En ocasiones solo disponemos de información parcial del secreto, lo que supone no conocer nada acerca de él. Para poder acceder a todo su contenido, entra en funcionamiento el concepto de umbral. En este sentido, se entiende que si alcanzamos un número determinado de particiones del secreto S ($t(S)$), del que existen $n(S)$ particiones de S, podemos llegar a la reconstrucción del mismo en su totalidad, y denotamos $r(s)$ como su umbral de reconstrucción.

Definición 4.2. Un esquema (t, n) de umbral es una disposición de acceso sobre un conjunto P que contiene n participantes con sus respectivas particiones del secreto S ($n(S)$), de manera que un conjunto de al menos t participantes con sus respectivas particiones del secreto S ($t(S)$), son capaces de recuperar el secreto. Sin embargo, en el caso que se junten k participantes con sus respectivas particiones del secreto S ($k(S)$), tales que $k(S) < t(S)$ el secreto S no se podrá recuperar.

4.1.3. Esquema de Shamir

El esquema de Shamir es uno de los primeros propuestos, es el ejemplo más famoso de un esquema de compartición de secretos. La estructura de acceso está formada por los subconjuntos con al menos t participantes de un conjunto con n participantes. Los esquemas con estructura de acceso de este tipo se denominan esquemas de umbral (t, n).

4.1.3.1. Introducción

Este esquema tiene como base la interpolación de polinomios para la distribución del secreto S entre n participantes. Necesitaremos t participantes para la reconstrución de S.

El esquema de Shammir consiste en ocultar información en un polinomio aleatorio. La utilización de este polinomio nos permite realizar particiones basadas en evaluaciones del polinomio. Dada cierta información parcial del polinomio, podemos llegar

a recuperar el secreto que estaba escondido en él. En el caso de que un participante recupera las suficientes particiones se podrá entonces recuperar el secreto completo.

Explicaremos a continuación el protocolo que propone Shamir para la distribución del secreto.

Sea El secreto S un número entero y t un número entero que pertenece $P = \{1, \ldots, n\}$tal que $1 \leq t \leq n$. Tómese un numero primo $q \geq \max\{S+1, n+1\}$. Se considera $q > n$, entonces el distribuidor D elige n elementos independientes y distintos de cero de $\mathbb{Z}_q$, que se denotan como x_i, con $1 \leq i \leq n$. Para $1 \leq i \leq n$, D asigna el valor x_i a P_i, de forma pública.

Cuando D comparte el secreto $S \in \mathbb{Z}_q$, toma al azar $t-1$ elementos de $\mathbb{Z}_q$, denotados como $a_1, \ldots, a_{t-1}$, estos elementos corresponden a los coeficientes que forman el polinomio, el cual debe ser de un grado menor o igual a $t-1$, de esta manera el coeficiente que contendrá el secreto siempre será a_0.

El polinomio requerido se muestra a continuación

$$f(x) = a_0 + \sum_{j=1}^{t-1} a_j x^j \in \mathbb{Z}_q[X]$$

D calcula $y_i = f(x_i)$, y otorga la partición y_i a P_i, es obvio que $a_0 = f(0)$.

4.1.3.2. Interpolación de Lagrange

Para la reconstrucción del secreto se utiliza la interpolación de Lagrange, debido a que es posible reconstruir polinomios utilizando varios puntos en los que se evalúa el polinomio.

Cada participante obtiene una serie de puntos. Se pueden juntar un conjunto de participantes lo suficientemente grande, pudiendo llegar a reconstruir el secreto. Antes de reconstruir el secreto, es necesario la reconstrucción del polinomio. La fórmula de la interpolación de Lagrange permite la reconstrucción del polinomio.

Definición 4.3. Interpolación polinomial de Lagrange:

Dado un conjunto de $k+1$ puntos

$$(x_0, y_0), \ldots, (x_k, y_k)$$

donde todos los x_j se asumen distintos, el polinomio interpolador en la forma de Lagrange es la combinación lineal:

$$L(x) = \sum_{j=0}^{k} y_j * \ell_j(x)$$

De bases polinómicas de Lagrange:

$$\ell_j(x) = \prod_{i=0, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} * \cdots * \frac{x - x_{j-i}}{x_j - x_{j-i}} * \frac{x - x_{j+i}}{x_j - x_{j+i}} * \cdots * \frac{x - x_k}{x_j - x_k}$$

Para todo $i \neq j$, $\ell_j(x)$ incluye en el numerador el término $(x - x_i)$, de modo que el producto completo valdrá cero en $x = x_i$.

$$\forall (j \neq i) : \ell_j(x_i) = \prod_{m\neq j} \frac{x_i - x_m}{x_j - x_m} = \frac{x_i - x_0}{x_j - x_0} * \cdots * \frac{x_i - x_i}{x_j - x_i} * \cdots * \frac{x_i - x_k}{x_j - x_k} = 0$$

Por otro lado,

$$\ell_j(x_j) = \prod_{m\neq j} \frac{x_j - x_m}{x_j - x_m} = 1$$

Todas las bases polinómicas de Lagrange valen cero en $x = x_i$, excepto $\ell_j(x)$, para el que aplica $\ell_j(x_j) = 1$, puesto que carece del término $(x - x_j)$ en el numerador.

Por tanto, se deriva que $y_j \ell_j(x_j) = y_j$, en cada punto x_j,

$$L(x_j) = y_j + 0 + 0 + \cdots + 0 = y_j,$$

demostrando que L interpola la función de forma exacta.

Nota 4.4. Si tenemos un polinomio L, donde conocemos la evaluación de algunos puntos, no buscaremos resolver el polinomio, sino que empezaremos a dividir el problema en varias partes y las resolveremos una a una.

Ejemplo 4.1. Explicaremos este procedimiento con un pequeño ejemplo numérico. Tomaremos $\mathbb{Z}_q = 11$ y un polinomio L que tiene los siguientes valores $y_1 = 2,\ y_2 = 4,\ y_3 = 3,\ y_4 = 4$ en los primeros 4 puntos y 0 en los demás puntos.

Tomamos $\ell_1, \ell_2, \ell_3, \ell_4$, uno por cada punto del polinomio L. Para reconstruir el polinomio, mostramos lo que haríamos con ℓ_1 y se seguirá el mismo procedimiento para $\ell_2,\ \ell_3,\ \ell_4$. Los polinomios tienen como valor 1 en un solo punto y en los demás puntos el valor 0. De tal manera que tomaremos $\ell_1(1) = 1, \ell_1(2) = 0, \ell_1(3) = 0$ y $\ell_1(4) = 0$.

Para la reconstrucción del polinomio L tomamos cada polinomio y le multiplicamos por el coeficiente correspondiente. Si sumamos todos nos dará como resultado:

$$2 \cdot \ell_1 + 4 \cdot \ell_2 + 3 \cdot \ell_3 + 4 \cdot \ell_4.$$

Podemos observar que la suma en el primer punto da como resultado $2 + 0 + 0 + 0$, lo mismo sucede en los demás puntos.

Entonces tenemos el polinomio $(x-2)(x-3)(x-4)$, y se visualizará que las evaluaciones de los ℓ_1 en los puntos son correctas y nos dan como resultado $\ell_1(2) = 0, \ell_1(3) = 0$ y $\ell_1(4) = 0$.Para que $\ell_1(1) = 1$, tenemos que dividir entre -6 ya que $(1-2)(1-3)(1-4) = -6$.

La fórmula para la construcción de ℓ_1 sería

$$\ell_1 = \prod_{i=1, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} \cdot \frac{x-4}{1-4} = \frac{(x-2)(x-3)(x-4)}{-6},$$

dando como resultado $\ell_1(1) = 1\ \ell_1(2) = 0, \ell_1(3) = 0$ o $\ell_1(4) = 0$. Y se puede construir $\ell_1, \ell_2, \ell_3, \ell_4$, de la misma manera.

$$\ell_2 = \prod_{i=2, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} \cdot \frac{x-4}{2-4} = \frac{(x-1)(x-3)(x-4)}{-6}$$

$$\ell_3 = \prod_{i=3, i\neq j} \frac{x - x_i}{x_j - x_i} = \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} \cdot \frac{x-4}{3-4} = \frac{(x-1)(x-2)(x-4)}{-6}$$

$$\ell_4 = \prod_{i=3, i\neq j} \frac{x - x_i}{x_j - x_i} = \frac{x-1}{4-1} \cdot \frac{x-2}{4-2} \cdot \frac{x-3}{4-3} = \frac{(x-1)(x-2)(x-3)}{-6}$$

Si tomamos los valores y_1, y_2, y_3 e y_4 podemos construir la fórmula de $L(0)$.

Por la interpolación de Lagrange $L(x) = \sum_{j=0}^{k} y_j * \ell_j(x)$ y la evaluación en 0 de los polinomios.

$$L(0) = y_1 \cdot \ell_1(0) + y_2 \cdot \ell_2(0) + y_3(0) \cdot \ell_3(0) + y_4 \cdot \ell_4(0) =$$

$$2 \cdot \ell_1(0) + 4 \cdot \ell_2(0) + 3 \cdot \ell_3(0) + 4 \cdot \ell_4(0) = 2 \cdot 4 + 4 \cdot 6 + 3 \cdot 4 + 4 \cdot 1 = 48 \quad \text{mód } 11 = 4$$

En los (ECS) sabemos que el grado del polinomio está acotado. El distribuidor compartirá el secreto eligiendo un polinomio de coeficientes aleatorios pero a los sumo de grado $t-1$. Entonces necesitaremos de t puntos para reconstruir $L(x)$.

Nota 4.5. Los esquemas públicamentes verificables $PVSS$, que comentamos en la figura 4.1 y detallaremos en el punto 4.2.1 utilizarán una versión simplificada de la interpolación de Lagrange. En lugar de recuperar el polinomio, intentaremos recuperar la evaluación en 0. Para ello tomaremos la siguiente fórmula:

$$\ell_j(x) = \prod_{i=0, i \neq j}^{k} \frac{x - x_i}{x_j - x_i}$$

$$\ell_j(0) = \prod_{i=0, i \neq j}^{k} \frac{0 - x_i}{x_j - x_i} = \prod_{i=0, i \neq j}^{k} \frac{x_i}{x_j - x_i}$$

Esta nota será utilizada en la votación electrónica, cuando demos el ejemplo numérico de este protocolo, punto 6.4.5.1.

4.1.3.3. Ejemplo usando el esquema de Shamir

Como hemos comentado al principio de la sección, las particiones dadas a cada participante se encuentran en $\mathbb{Z}_q$. El distribuidor D construye un polinomio aleatorio $f(x) \in \mathbb{Z}_q[x]$ que tiene como grado máximo $t-1$. Cada participante P_i obtiene un par (x_i, y_i) del polinomio.

Para explicar mejor este esquema trataremos un ejemplo numérico muy sencillo que nos permitirá entender el protocolo a la perfección.

Ejemplo 4.2. Enunciado: Supongamos que queremos compartir un dato que corresponde a $S = 10$ y tomamos $\mathbb{Z}_q = 11$. El secreto S se desea dividir en 6 partes $(n = 6)$; de forma que cualquier subconjunto $(t = 3)$ sea suficiente para reconstruir el secreto, entonces estamos describiendo un esquema de umbral $(3, 6)$.

Al azar se obtienen $t-1$ valores: por ejemplo 120 y 89. Se usan coeficientes que definen estos valores ($a_1 = 120 \mod 11 = 10$; $a_2 = 89 \mod 11 = 1$), mientras que

el coeficiente que corresponde al secreto se define como $a_0 = S = 10$.
Se describe el polinomio en un orden especifico, es decir

$$f(x) = a_0 + a_1 x + a_2 x^2$$

$$f(x) = 10 + 10x + x^2$$

Posteriormente dado que $n = 6$, se calculan seis puntos a partir del polinomio, es decir que evaluamos el polinomio cuando $x = 1, 2 \ldots 6$, de manera que se obtienen puntos de la forma $(x_i, f(x_i))$.
$(1, f(1)); (2, f(2)); (3, f(3)); (4, f(4)); (5, f(5)); (6, f(6))$, evaluando estos valores en el polinomio, se obtienen los puntos:

$$(1, 10); (2, 1); (3, 5); (4, 0); (5, 8); (6, 7)$$

A cada participante le corresponde un único punto x y $f(x)$.

Reconstrucción: como se describió anteriormente, para reconstruir el secreto bastará con tres puntos $(t = 3)$. Considere 3 puntos cualesquiera

$$(x_0, y_0) = (2, 1); (x_1, y_1) = (3, 5); (x_2, y_2) = (5, 8)$$

Usando la interpolación polinómica de Lagrange se tiene:

$$\ell_0 = \frac{x - x_1}{x_0 - x_1} * \frac{x - x_2}{x_0 - x_2} = \frac{(x-3)}{(2-3)} * \frac{(x-5)}{(2-5)} = (\frac{1}{3}x^2 - \frac{8}{3}x + 5) \quad \text{mód } 11 = \frac{1}{3}x^2 + x + 5$$

$$\ell_1 = \frac{(x - x_0)}{(x_1 - x_0)} * \frac{(x - x_2)}{(x_1 - x_2)} = \frac{(x-2)}{(3-2)} * \frac{(x-5)}{(3-5)} = (-\frac{1}{2}x^2 + \frac{7}{2}x - 5) \quad \text{mód } 11 = 5x^2 + \frac{7}{2}x + 6$$

$$\ell_2 = \frac{(x - x_0)}{(x_2 - x_0)} * \frac{(x - x_1)}{(x_2 - x_1)} = \frac{(x-2)}{(5-2)} * \frac{(x-3)}{(5-3)} = (\frac{1}{6}x^2 - \frac{5}{6}x + 1) \quad \text{mód } 11 = \frac{1}{6}x^2 + x + 1$$

Por lo tanto,

$$f(x) = \sum_{j=0}^{k-1} y_j * \ell_j(x)$$

$$= 1 * (\frac{1}{3}x^2 + x + 5) + 5 * (5x^2 + \frac{7}{2}x + 6) + 8 * (\frac{1}{6}x^2 + x + 1)$$

$$= 10 + 10x + x^2$$

Teniendo en cuenta que el secreto es el coeficiente a_0, entonces el secreto original es $S = 10$.

4.2. Esquemas de compartición de secretos verificables (VSS)

En los esquemas de compartición de secretos de umbral se asume que todos los participantes que participan en este protocolo actúan de manera honesta. En cambio, en los esquemas de compartición de secretos verificables (VSS), los participantes pueden intentar obstaculizar el proceso, por eso se requiere que los participantes se encarguen de demostrar que son totalmente honestos.

Los participantes deshonestos pueden comportarse de diferentes maneras:

a) Un distribuidor que envía particiones incorrectas a una serie de participantes, o a todos.

b) Participantes que envían particiones incorrectas durante el proceso de reconstrucción, para entorpecerlo.

En los esquemas VSS el distribuidor intenta proteger la información, por eso envía a cada uno de los participante los datos por una serie de canales privados. El objetivo de los esquemas VSS es sobrevivir y prevenir los ataques de los participantes deshonestos, que intentan producir errores en el protocolo.

Existen diversas técnicas para realizar el proceso de verificación, una de las más utilizada es la prueba de conocimiento cero 3.2.1. Tanto en la fase de distribución como en la reconstrucción se exigen pruebas a los participantes de la corrección de las particiones. Este esquema resuelve el problema de los ataques de los participantes deshonestos. Por este motivo se usan en la subasta y la votación electrónica.

Solo los participantes pueden realizar las pruebas, pero sería ideal que cualquier participante las verifique. Por este motivo, surgen los siguientes esquemas en los que centraremos nuestro estudio.

4.2.1. Esquemas de compartición de secretos verificables publicamente (PVSS)

4.2.1.1. Introducción

Los esquemas de compartición de secretos verificables públicamente $(PVSS)$ son una extensión de los esquemas VSS. Estos esquemas tienen como principal característica que los participantes pueden verificar sus propias particiones, además

cualquier participante puede verificar que recibió estas particiones de forma correcta.

Los problemas que pueda ocasionar el distribuidor, explicados en (a), se solventan al verificar públicamente sus propias particiones. Los problemas que surgen con los participantes en (b) pueda tratarse de forma implícita, de manera que los participantes se encarguen de liberar sus particiones.

Como las particiones pueden ser verificadas por cualquier participante, en los esquemas $PVSS$ se realiza la comunicación a través de canales públicos autenticados, utilizando cifrado de clave pública. Entonces el secreto solo se encuentra oculto computacionalmente.

El esquema de compartición de secretos de Shamir es la base de este tipo de esquemas. El problema del logaritmo discreto es la primitiva de seguridad para este protocolo. Esto implica que si encontramos un algoritmo eficiente para resolver el problema de DL, entonces el protocolo $PVSS$ será inseguro. El esquema $PVSS$ es utilizado en todo grupo para el que el problema del logaritmos discreto no se pueda aplicar. La prueba de conocimiento cero en el protocolo PVSS permite verificar la exactitud de los datos.

La construcción $PVSS$ es muy sencilla ya que los protocolos constan de pocos pasos. Nosotros explicaremos los esquemas $PVSS$ de manera general y no interactiva. La estructura general que sigue esta serie de esquemas, es la siguiente:

Todos comienzan con una **fase de inicialización** en la que los participantes se registran y tienen una clave pública asociada. Mas adelante nos encontramos con una **fase de distribución** en la que el distribuidor es el encargado tanto de crear las particiones, como de publicarlas de manera encriptada y por supuesto de publicar una prueba de ellas.

Además de estas dos fases todos estos esquemas tienen una **fase de verificación** en la que cualquier participante que conozca la clave pública puede verificar las particiones. Como medida de seguridad en la fase de verificación, si la prueba falla, el distribuidor falla y el protocolo es abortado.

Por último la **fase de reconstrucción** consiste en una fase de descifrado y la puesta en común de las particiones. Como cada participante contiene su clave pública, los participantes son los encargados de descifrar sus propias particiones. Además, los participantes también son los encargados de publicar la prueba, y estas sirven como corte para poder excluir a los participantes deshonestos. Para acabar el protocolo, se permite reconstruir el secreto por la unión de un número determinado de participantes con

sus correspondientes particiones, y pleno conocimiento de las claves públicas.

4.2.1.2. Fases del protocolo

En este punto explicaremos de forma más detallada todo las fases del protocolo. Los participantes saldrán o entrarán en el sistema de forma dinámica, para poder entrar y salir del sistema solo necesitan una clave pública registrada.

1. Inicialización: Esta fase se realiza sin interacción entre el distribuidor y los participantes, y es la encargada de generar los parámetros. Cada participante P_i registra una clave pública para ser utilizada con un método de cifrado de clave pública E_i. En los esquemas PVSS participan un subconjunto de los participantes registrados $P_1, \ldots, P_n$.

2. Distribución de las particiones: La distribución es realizada por un distribuidor D de un secreto $s \in \Sigma$. Lo primero que hace el distribuidor es producir para P_i con $i = 1, \ldots, n$ el recurso compartido cifrado $E_i(s_i)$. Además el distribuidor publica una cadena $PROOF_D$ que garantiza la reconstrucción del valor s y se utiliza para mostrar que cada E_i cifra un s_i compartido.

3. Verificación de las particiones: Cualquiera que tenga conocimiento de las claves públicas que se utilizan en los métodos de cifrado, puede verificar los secretos. Para ver que E_i es un cifrado correcto para cada participante P_i, cualquier participante puede realizar las verificaciones, entonces en caso de que falle una o varias verificaciones, se cancelaría el protocolo y diríamos que el distribuidor ha fallado. No obstante, el protocolo podría continuar y pensar en ello como un esquema umbral $(t, n - c)$, donde c será el número de verificaciones que resultaron erróneos.

4. Reconstrucción:El protocolo consta de dos pasos:

4.1. Descifrado de las particiones.

No todos los participantes logran descifrar las particiones s_i de E_i, solo el conjunto de participantes que tenga conocimiento de las clave públicas. La cadena $PROOF_{P_i}$ es la encargada de mostrar que el recurso liberado compartido es correcto.

4.2. Puesta en común de las particiones.

Las cadenas $PROOF_{P_i}$ son utilizadas para conseguir detectar y expulsar a los participantes que no han actuado de manera correcta. La creación de los secretos se puede

hacer a partir de las particiones de cualquier conjunto de participantes, para ello los participantes deben acreditar una prueba del correcto descifrado. Entonces, cualquier parte puede resolver la particiones correctas y agruparlas.

Este esquema reduce la interacción entre los participantes, cosa que no sucede de la misma manera en los esquemas VSS. En los esquemas VSS no interactivos se pueden presentar quejas si se percibe cualquier fallo en el proceso. Estas quejas deben ser resueltas para la comprobación de si el reparto del secreto ha funcionado de manera correcta. En cambio, PVSS no interactivo no tiene esta interacción, y cualquier participante puede verificar la salida del distribuidor.

Nota 4.6. Los esquemas $PVSS$ se encargan de filtrar información parcial del secreto, teniendo uso en importantes aplicaciones como el voto electrónico de un solo bit $(0$ o $1)$. El ejemplo numéricos más detallado de este proceso lo daremos en la sección 6.4.5.1 una vez que hayamos explicado la votación electrónica.

4.3. Compartición de secretos homomórficos

Teniendo en cuenta la definición de homomorfismo 4.3, la compartición de secretos homomórficos nos permitirá realizar operaciones sobre los datos cifrados, dando como resultado información cifrada. En ningún momento del proceso se revela información. Entonces a la hora de manejar datos, se observa que para la manipulación de los datos originales, existe la correspondiente manipulación de los datos transformados.

La noción de compartir secretos homomórficos es muy relevante en particular para el voto electrónico. Un esquema de cifrado homomórfico es un sistema de cifrado que nos va a permitir realizar cálculos sobre los datos, sin la necesidad de descifrarlos. La utilización de los esquemas PVSS nos permite juntar particiones de secretos independientes de tal manera que la reconstrucción de las particiones combinadas dan como resultado un secreto combinado.

En los esquemas PVSS, hay tanto una operación $\bigotimes$ en las particiones, como en las particiones encriptadas $\bigoplus$. Entonces para todos los participantes se cumple que $E_i(s_i) \bigotimes E_i(s_i') = E_i(s_i \bigoplus s_i')$. Al descifrar los recursos compartidos cifrados, el secreto recuperado será igual a $s \bigoplus s'$. El esquema de compartición de secretos generado $\bigoplus$ es homomórfico.

4.4. Esquemas de compromiso

En las pruebas de conocimiento cero 3.2.1, hemos visto que el primer mensaje que genera el Probador es un compromiso con el Verificador. Este compromiso sirve para garantizar que el proceso se realice de manera totalmente segura.

Este tipo de esquemas tratan de comprometer un valor que se mantiene de forma anónima, para más tarde ser revelado. Tienen numerosas aplicaciones, entre ellas la utilización de este tipo de esquemas en pruebas de conocimiento cero. Con la utilización de estos esquemas se garantiza que no se revele ni aprenda ningún tipo de información durante la realización del proceso, solo el resultado final de este. Protocolos como la subasta o comercio electrónico utilizan estos esquemas para evitar la asincronía de la red.

Podemos definir estos esquemas de manera general como un método en que se permite enviar información secreta sin posibilidad de que esta se modifique durante el proceso. Tampoco se revela ningún dato mientras que el valor se mantiene oculto. En estos esquemas el receptor solo ve la información cuándo el emisor revela el mensaje. Para ello el emisor otorga al receptor la "llave" con la que observará el mensaje, totalmente secreto hasta ese momento.

Esquemas de compromiso protocolo.

Los esquemas de compromiso tienen dos fases, una primera fase de compromiso y una segunda de apertura. Tiene como participantes dos entidades que llamaremos A y B.
En la **fase de compromiso** una entidad A se compromete con la entidad receptora B sobre un valor que no se podrá cambiar siempre que se mantenga el valor oculto. El contenido se encuentra oculto al receptor hasta la fase de apertura. El emisor A en esta fase genera una clave pública c_k para comunicarse con el receptor. El emisor quiere transmitir el mensaje m al receptor, de tal manera que se genera el compromiso $Com(m, c_k) = (c)$ y el mensaje cifrado d.

El emisor A en la **fase de apertura** revela el mensaje al receptor B haciéndole llegar (c, d), es decir, el compromiso y el mensaje cifrado, para que pueda abrir el mensaje. El compromiso c no debe revelar ninguna información sobre el valor que "esconde". El receptor obtiene utilizando (c, d) y la clave pública c_k el mensaje m o un mensaje de error, $m = Apertura((c, d), c_k)$. En los esquemas de compromiso el emisor A no puede cambiar el mensaje y el receptor no puede tener ninguna información acerca del mensaje.

A la hora de ver la seguridad de estos esquemas, el compromiso c no debe revelar ninguna información sobre el mensaje que oculta, y nosotros teniendo la clave c_k no podemos construir dos mensajes m_0 y m_1 de tal manera que sus compromisos c_0 y c_1 puedan ser diferentes. No es posible deshacer un compromiso. Un adversario puede construir la terna (c, d, d') a la que llamaremos colisión, de modo que si (c, d) es un compromiso válido para m, y (c, d') al mismo tiempo es un compromiso válido para m' de tal manera que $m \neq m'$.

Ejemplo 4.3. Una aplicación de este tipo de esquemas es la subasta o el comercio electrónico. Si tenemos un comprador A y un vendedor B, el comprador A no quiere gastarse más que el precio de compra denominado a, por el contrario, el vendedor B desea vender su producto por un precio mayor que el precio de venta denominado a. Las cantidades a y b se guardan en secreto para no favorecer a su oponente. Entonces A y B acuerdan un protocolo para que el producto sea vendido al precio medio $p = \frac{(a+b)}{2}$. El esquema de compromiso se encarga dar una solución a este problema.

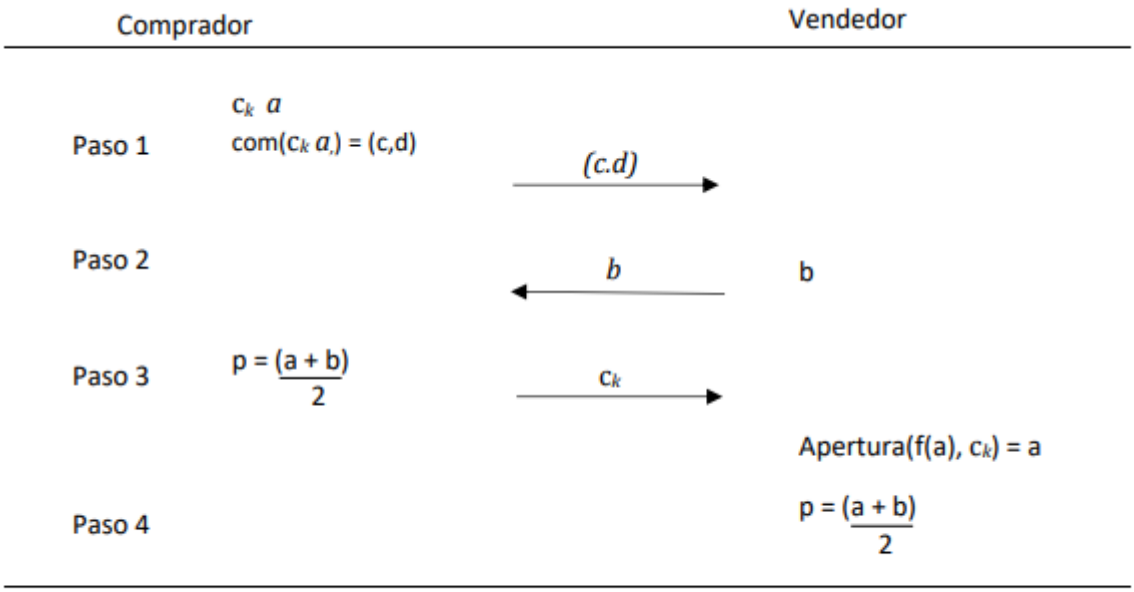

Imagen 4.2: Ejemplo de esquema de compromiso, comercio electrónico.

En el **Paso 1** el comprador genera una clave pública c_k y comunica a B el compromiso c, y el valor oculto a y d.

En el **Paso 2** el vendedor B envía a A el valor b.

En el **Paso 3** el comprador envía a B su clave pública y calcula $p = \frac{(a+b)}{2}$.

En el **Paso 4** al recibir la clave pública el vendedor realiza la apertura del compromiso con los valores recibido anteriormente (c, d), Apertura$((c, d), c_k)$. Para finalizar, B calcula $p = \frac{(a+b)}{2}$ y ambos ven que el valor p que obtuvieron es el mismo.

Este ejemplo de comercio electrónico puede extenderse de manera más compleja a una subasta electrónica, como veremos más adelante, en la sección 7.

4.4.1. El secreto comprobable de Pedersen

El esquema de compromiso de Pedersen es un ejemplo de compromiso homomórfico basado en el problema del logaritmo discreto. Como todo esquema de compromiso tiene dos fases: la fase de compromiso y la fase de apertura. En el proceso participan dos entidades A y B.

Este tipo de esquema trabaja con elementos del subgrupo de $\mathbb{Z}_p^*$ con orden q, donde p y q son números primos grandes y q divide $p-1$. Sea el generador $g \in \mathbb{Z}_p^*$ donde el problema del logaritmo discreto es difícil de solucionar, en la **fase de compromiso** la entidad A genera una clave pública $c_k := (p, y, g)$ para comunicarse con el receptor, donde y un elemento aleatorio de $\mathbb{Z}_{p-1}$ de modo que no se conoce $log_g\ y$.
A recibe como entrada un bit b y elige al azar un exponente $r \in \mathbb{Z}_p^*$, generando el compromiso $Com(r, b) = g^r y^b \mod p$, entonces $d := r$.

En la **fase de apertura** el emisor A revela el compromiso al receptor B haciéndole llegar (c, d). B comprueba si $c = g^r y^b$ para $b = 1$ o $b = 0$. En caso de no cumplirse devuelve error.

La seguridad de este esquema se basa en que tanto r como y son elementos aleatorios, lo que supone que si un adversario construye (c, r_0) y (c, r_1) tal que: Apertura$(c, r_0) = 1$ y Apertura$(c, r_1) = 0$. El problema del logaritmo discreto ha sido resuelto ya que $g^{r_0} = g^{r_1} y$ y por tanto $y = g^{r_0 - r_1}$.

Este esquema es una ampliación del esquema de compartición de secretos de Shamir y los propios participantes puedan verificar las particiones utilizando los

esquemas VSS de la siguiente manera:

1. A construye un polinomio $f(x) = \sum_{j=0}^{T-1} a_k x^k$, donde a_0 es el secreto que se desea compartir y a_k con $k = 1, 2, \ldots, T-1$ números enteros aleatorios.
Si usamos un segundo polinomio del mismo grado que $f(x)$, $f'(x) = \sum a'_k x^k$, para a'_k con $k = 1, 2, \ldots, T-1$ números enteros aleatorios. Se puede generar los compromisos $E_0, \ldots, E_{t-1}$, para probar que las particiones del resto de usuario fueron correctamente calculadas.

2. A publica $E_j = g^{a_k} y^{a'_k}$ para $k = 0, 1, \ldots, T-1$.

3. A para compartir a_0 enviará al participante $j-$ésimo el par $(f(j), f'(j))$ como partición al participante P_i.

4. P_i verifica $g^{f(j)} y^{f'(j)} = \prod_{k=0}^{T-1} {E_k}^{j^j}$ mód p. Si se pasa la verificación, P_i puede estar seguro de que $log_g \ E_0$.

Las propiedades que tiene este esquema de compromiso son:

- Corrección: si el propietario del secreto utilizó el esquema VSS, puede compartir su secreto de tal manera que el recurso compartido pase la verificación.

- Solidez: Los valores E_k no ofrecen ninguna información del resto de participantes pero si permiten verificar que los elementos $f(j)$ y $f'(j)$ son evaluaciones de polinomios de grado $t-1$. Si al menos T participantes obtienen las particiones correctas, se puede recuperar $log_g \ E_0$ por ellos colectivamente.

- Homomorfismo: si compartimos múltiples secretos entre los mismos conjuntos de particiones de los participantes, sumando las particiones, se puede recuperar la suma de los secretos. El esquema de compromiso de Pedersen posee la propiedad homomórfica aditiva, si tomamos $Com\ (r, b) = g^r y^b$ mód $p, [Com\ (r, b) \cdot Com\ (r', b')] = com\ (r + r', b + b')$. Esta propiedad es aplicable en el protocolo de votación electrónica, los votantes colocan sus votos en compromisos homomórficos y a la hora de realizar el recuento, los votos y votantes se cuentan cogiendo el producto de todos los compromisos.

Capítulo 5

Computación Segura entre múltiples partes (MPC)

5.1. Introducción

La computación segura entre múltiples partes (MPC) es un concepto de seguridad criptográfica que utiliza los esquemas de compartición de secretos (ECS), permitiendo que varias partes que no confían entre si puedan calcular conjuntamente los recursos del sistema. Los resultados si pueden ser memorizados, pero no las aportaciones de las demás partes. El objetivo es calcular de forma segura un problema acordado, en el que cooperan n participantes. Cada participante realiza una tarea de cálculo basada en los datos privados que cada uno proporciona y ningún otro participante puede calcular.

El concepto *MPC* fue introducido por Andrew C. Yao. en 1982 y ejemplificado por el "Problema de los millonarios [Yao]": “Dos millonarios quieren saber quién es más rico; sin embargo, no quieren descubrir inadvertidamente cualquier información adicional sobre la riqueza de cada uno. ¿Cómo pueden llevar a cabo una conversación así? ". Se quiere saber quien es el más rico de los dos millonarios, sin descubrir en ningún momento el dinero que tiene cada uno.

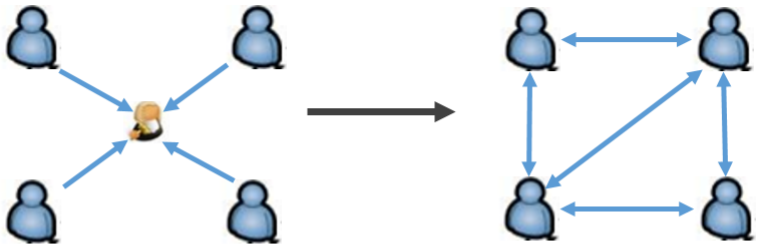

Imagen 5.1: Relación MPC

Este protocolo no usa medios convencionales, en la que partes comunes podrían acumular información convencional, sino que existe un participante que se encarga de recopilar las informaciones secretas de los demás participantes. Este juez confidencial calcula el resultado de todas las entradas y revela únicamente una salida. Pero no siempre tiene que existir un participante de confianza, y mediante el uso computacional entre distintas partes pueden lograr el mismo nivel de confianza. Si no existe una figura de confianza tenemos que tener en cuenta que la capacidad para validar las entradas por el protocolo MPC se basa en la honestidad de los participantes.

Consideremos n partes $P_1, \ldots, P_n$ que tienen como entradas privadas $x_1, \ldots, x_n$, y que desconfían entre si. Si se quiere calcular una función $y = f(x_1, \ldots, x_n)$ solo filtrando su salida, se debe mantener siempre la privacidad de las entradas. El objetivo de MPC es permitir esto de forma que solamente se precise la comunicación entre las partes.

Hay dos tipos de protocolos MPC, los diferenciaremos según la corrupción de la información. Existen los protocolos MPC que están a salvo de la corrupción pasiva, en la que todos los participantes son honestos y manejan datos correctos. Además, existen los protocolos MPC que son seguros contra corrupción activa evitando el envió de datos incorrectos. En este caso se utilizan los protocolos de compartición de secretos verificable (VSS) y la prueba de conocimiento cero. Se permite a los participantes involucrados verificar que sus particiones sean consistentes, lo que significa que el protocolo VSS podrá detectar particiones incorrectas.

MPC ha ido creciendo a lo largo del tiempos siendo cada vez un protocolo más eficiente, llegándose a usar en una gran variedad de problemas. MPC se puede utilizar tanto para generar claves y calcular operaciones criptográficas. Se intenta repartir las

particiones en diferentes entornos, complicando a los adversarios su obtención.

5.2. Adversarios

Es importante saber que no se puede confiar en todos los participantes en el protocolo *MPC*, dado que algunos participantes pueden transformar la información o intentar interrumpir el proceso. Dividimos estos adversarios en dos grupos principales dependiendo de la manera que transformen la información. Si en el cálculo del protocolo existen n partes involucradas, algunas pueden comportarse sin entorpecer el protocolo (partes honestas) y algunas pueden querer romper la privacidad de las entradas de las otras partes (partes corruptas).

Nota 5.1. El protocolo no sabe qué partes son corruptas y que partes son honestas. Por lo tanto, el objetivo será asegurarse de que las partes corruptas no se enteren de las partes honestas. Cualquiera de las partes puede ser corrupta, por eso a priori no se puede confiar en ninguna.

El protocolo *MPC* debe mantener la privacidad de las entradas ante un adversario que se encarga de controlar un subconjunto de las partes y de corromperlas. Sin embargo la pregunta es qué poder tiene el adversario al controlar una de las partes. El adversario se encarga de recopilar información, cuanta más información tenga más poderoso será. Puede instruir a las partes bajo su mandato, tomando las decisiones de cuándo y cómo actuar para conseguir que el protocolo se desestabilice. Lo más importante es saber en todo momento hasta donde pueden llegar a entorpecer el protocolo las partes corruptas.

Las partes corruptas y lo poderoso que es el adversario computacionalmente tienen una gran importancia en el desarrollo del protocolo. Hay distintos tipos de adversarios:

- **Adversarios pasivos**: Se encarga de obtener el estado interno de todas las partes corruptas. Las partes corruptas siguen con total normalidad el proceso del protocolo, y al tener constancia del estado interno de estas partes también se tiene conocimiento de todos los mensajes recibidos. Este adversario es bastante frágil, y si se previene a este tipo de ataques, el protocolo no tendrá fuga de información.

- **Adversarios activos**: Se encarga de que las partes corruptas que tiene bajo su mandato, se desvíen de su ejecución normal, entorpeciendo así el proceso

del protocolo. Es importante prevenir este tipo de adversarios ya que esto provoca que no importe cómo se comporten las partes y siempre se guardará la privacidad de las entradas de las partes honestas.

Cuando la corrupción es pasiva se obtiene acceso a información que no se debería tener derecho de acceder. En este caso podemos poner como ejemplo, un votante que pregunta y contrasta su voto con el de otro votante obteniendo de esa manera más información sobre el protocolo. Esta corrupción se puede prevenir con la utilización de los (ECS), porque el esquema garantiza que si t participantes no son honestos, entonces no podrán obtener ninguna información.

Por otro lado, si la corrupción es activa el adversario activo intenta enviar datos que no se deberían de enviar, para que las partes se desvíen del protocolo. Los esquema de compartición de secretos verificables públicamente $(PVSS)$ previenen este tipo de ataques. La validez de las particiones no solo es verificada por los participantes que las reciben sino que puede ser verificado por cualquier parte que interviene en la comunicación.

Número de partes corruptas.

Los adversarios se pueden visualizar como una entidad maestra que controla un subconjunto de las partes, cuantas más partes pueda corromper el adversario, mayor poder tendrá y le resultará más sencillo romper la privacidad de los demás participantes honestos.

Nota 5.2. El número de partes corruptas y su comportamiento son los parámetros más relevantes para tener consciencia del poder que puede tomar el adversario.

Si denotamos a n como el número total de particiones, y t la mayor cantidad de partes que pueden estar corrompidas, cuya identidad no es conocida. Por tanto tenemos diversas opciones que se pueden dar en el protocolo:

- $t < n$. Es el escenario más fuerte posible, mantiene la privacidad del protocolo. Se conoce este escenario como mayoría deshonesta.

- $t < n/2$. Se conoce este escenario como mayoría honesta. El adversario solo puede corromper a una minoría de las partes. En este caso, la privacidad del protocolo se rompe tan pronto como el adversario corrompa a la mitad o más de las partes. La mayoría honesta permite protocolos más eficientes con garantías de seguridad más sólidas, asumiendo que las partes tienen acceso a los canales

de comunicación.

- $t < n/3$. El adversario solo puede corromper a un tercio de las partes. Este escenario garantiza una seguridad aún mejor y una mayor eficiencia. Además, puede tener sentido cuando n es muy grande, asumiendo que los canales son privados y autenticados.

5.3. Seguridad de MPC

El objetivo del protocolo *MPC* es permitir una computación distribuida de manera segura. Sabemos que en la ejecución del protocolo se puede interponer un adversario. El resultado de este ataque siempre tiene como intención obtener información o bien que el resultado del cálculo sea incorrecto.

Ahora mostraremos todas las propiedades que debe cumplir el protocolo *MPC* para ser totalmente seguro:

1. Privacidad: Las partes de un protocolo no deben aprender más información que su propia salida prescrita. En concreto solo se puede aprender la información que se deriva de la salida. Por ejemplo, en la subasta electrónica la oferta revelada es la del licitador que ha resultado ganador, ya que las demás ofertas fueron más bajas que esta. No se puede revelar el precio exacto de las ofertas perdedoras.

2. Corrección: Esta propiedad garantiza a todas las partes que la salida es correcta. Entonces el adversario intenta desviar el resultado a calcular por las partes del protocolo, esto provoca que si existe una corrupción en el proceso, no se pueda cambiar el resultado.
En el ejemplo de la subasta electrónica se garantiza que la oferta más alta es la ganadora y ningún participante puede hacer nada para cambiarlo, ni el subastador.

3. Independencia de los bienes: Las partes corruptas tienen que elegir sus bienes independientes de las demás partes. Parece que esta independencia está integrada en la privacidad, pero no es así. Si seguimos en el ejemplo de la subasta sellada, se deben fijar las ofertas independientes del valor de las demás ofertas existentes, y sería posible que se generé una oferta más alta sin tener conocimientos de la oferta original.

4. Entrega de resultados garantizados: El adversario no puede impedir que las partes honestas reciban su salida. Por lo tanto, el adversario no puede realizar un ataque de denegación de servicios.

5. Equidad: Si las partes honestas reciben sus salidas, entonces las partes corruptas recibirán sus resultados. Tenemos que tener en cuenta que la entrega de la salida implicará justicia.

Si el protocolo es totalmente seguro tiene que resistir a cualquiera de los ataques que realizaran los adversarios. Podemos definir seguridad como el cumplimiento de todas estas propiedades, pero también podemos ver la seguridad como el resultado de la ejecución del protocolo de manera ideal. Si se ejecuta de manera real un protocolo se intentará que funcione de manera ideal, esto se le llama el paradigma de simulación ideal/real.

Nota 5.3. En un cálculo *MPC* seguro, el protocolo no se ejecuta de manera aislada, sino que forma parte del propio sistema. Se suele utilizar un modelo modular en el que se crean protocolos más grandes a partir de protocolos más pequeños, utilizando *MPC* para llevar a cabo los cálculos.

La seguridad de una aplicación depende de que las partes utilicen las entradas correctas. Y hay que tener en cuenta que *MPC* no asegura el proceso. Es cierto que el cálculo que se realiza no revela ningún tipo de información, pero la salida de la función que se está calculando puede revelar información sensible. *MPC* no resuelve la privacidad y su función es asegurar el proceso de computación.

La definición mostrada de seguridad no contempla el éxito de un adversario. Cabe preguntarse si el protocolo puede ser seguro bajo esta definición. Podemos asegurar que cualquier tarea de computación distribuida puede ser calculada de forma segura, sin que los adversarios puedan entorpecer el protocolo.

5.4. Ejemplo protocolo MPC

La pregunta que nos tenemos que hacer cuando pensamos en el protocolo *MPC*, es la siguiente: ¿cómo es posible que las partes calculen una función sin filtrar los datos, y preservando la privacidad?.

Consideremos que nos encontramos en $\mathbb{Z}_5$, y tenemos dos participantes Ana y Bob, que tienen como entrada x e y respectivamente. Ana y Bob quieren calcular la siguiente función:

$$z = f(x, y) = (xy) * (x + y) \quad \text{mód } 5.$$

Estos valores se encuentran en el conjunto $\{0, 1, \ldots, 4\}$. El objetivo es realizar esto de manera que Bob aprenda la entrada x de Ana, y a su vez Ana aprenda la entrada y de

Bob, además de lo que ya fue filtrado por la salida z.

Cualquier participante puede notar que $z = x^2 - y^2 \mod 5$. Como Bob sabe y, aprender z le permite aprender $x^2 \mod 5$, que filtra bastante sobre x, lo mismo ocurriría con Ana e y. Tenemos que tener en cuenta en primer lugar, que esta operación no proporciona x en su totalidad pero si sabe que cada cuadrado módulo 5 tiene dos raíces, una en el conjunto $\{1, 2\}$ y otra en el $\{3, 4\}$. En segundo lugar, no nos importa lo que filtre z sobre las aportaciones de las partes, dado que la definición de seguridad para *MPC* requiere que el adversario no aprende nada acerca de las entradas de las partes honestas, más allá de lo que ya se filtró desde la salida. Debemos saber que no se pueden ocultar las entradas si la salida las filtra.

Ocultar los datos

Necesitamos diseñar un protocolo que permita calcular $z = (xy) * (x + y) \mod 5$ y que no filtre nada más. Si observamos la función z parece que para obtener z necesitamos conocer las entradas x e y. La operaciones módulo 5 que se realizan en este protocolo son:

1. Calcular $u = x + y$ y $v = xy$.

2. Multiplicar $z = u * v$ y devuelve z.

Para ocultar los datos se pueden utilizar diferentes herramientas criptográficas como los esquemas de cifrado homomórficos (sección 4.3). Este esquema permite que los datos se transformen en una representación .ºculta" y aún así realizar las operaciones descritas anteriormente sin revelar ningún tipo de información. Si ciframos las entradas por un esquema de cifrado homomófico $Hom(x)$ y $Hom(y)$, entonces se pueden realizar los pasos anteriores, ya que que estos esquemas nos permiten realizar operaciones sin revelar información.

1. Ana cifra x como $Hom(x)$ y Bob cifra y como $Hom(y)$.

2. Calcular $Hom(u) = Hom(x) + Hom(y)$ y $v = Hom(x) - Hom(y)$

3. Multiplicar $Hom(z) = Hom(u) * Hom(v)$

4. Descifrar z y devolver z.

El objetivo de este diseño es calcular una función de forma segura, sin que no se filtre nada sobre los valores x e y mientras se están procesando. Aunque Ana y Bob en

este ejemplo aprenden z, el esquema homomórfico no es tan eficiente como pueden ser los (ECS) (capitulo4.1). Los (ECS) son utilizados en el protocolo MPC, pues las partes pueden comunicarse y ayudarse mutuamente a realizar el cálculo. La utilización del protocolo MPC unido a los (ECS). Lo ejemplificaremos numéricamente al explicar el protocolo de votación y subasta electrónica

Capítulo 6

Votación electrónica

Todos nos hemos encontrado en la tesitura de tener que emitir un voto para una elección, ya sea un tema simple y rápido, o un tema mucho más complejo como unas elecciones gubernamentales. A la hora de emitir un voto de manera electrónica, lo realizaremos desde un dispositivo, y lo más importante es que el proceso sea totalmente seguro.

6.1. Introducción

Vamos a comenzar esta sección describiendo los conceptos de voto electrónico, teniendo siempre muy presente las demandas de este protocolo y los requisitos de seguridad que tiene que cumplir un protocolo seguro.

La grabación electrónica directa de votación (DRE), es un dispositivo de votación electrónica que se está desarrollando en los últimos tiempos. Este sistema permite la realización de la votación electrónica y la aceleración en el conteo de votos. Un sistema de estas características siempre tiene que contener un software específico para realizar este procedimiento. Hay que tener en cuenta que es complicado que los participantes tengan una especialización en el proceso, ya que es algo muy novedoso. Esto unido a el posible ataque de adversarios dificulta bastante el proceso.

Los votos emitidos en un DRE se pueden realizar de diferentes formas. Nosotros diferenciaremos dos grandes bloques dependiendo de la manera de realizar el voto:

1. Presencial: De manera presencial la emisión de votos se pueden realizar de diferentes formas, pero todas ellas siguen el mismo procedimiento. El dispositivo de votación electrónica se encuentra conectado a un servidor. El servidor es la autoridad principal que llevará a cabo el conteo totalmente controlado de los votos.

Al igual que cualquier sistema de votación, el participante se tiene que identificar de alguna manera. Este registro se le permite realizar ya sea antes o durante el proceso de emisión del voto. Los dispositivos electrónicos también estarán en línea, lo que provocará que la autenticación pueda realizarse en el lugar físico de la votación.

2. Forma remota: También existe la opción de emitir los votos en un *DRE* de manera totalmente remota teniendo conexión a Internet, siempre identificándose con algún tipo de autenticación.

Ahora desarrollaremos todos los componentes que forman parte del proceso de votación:

- Votante: Es el participante que tiene todos los privilegios para poder realizar su propia elección sobre el tema a tratar.
- Autoridad de registro: Es el participante que se encarga de registrar a todos los votantes que pueden participar en el sistema electoral.
- Autoridad de recaudación: Es el participante que se encarga de recapitular todos los votos, puede que sea un sistema que funcione de manera electrónica.
- Autoridad de recuento: Es el participante responsable de realizar el conteo de votos y de dar el resultado de ello.

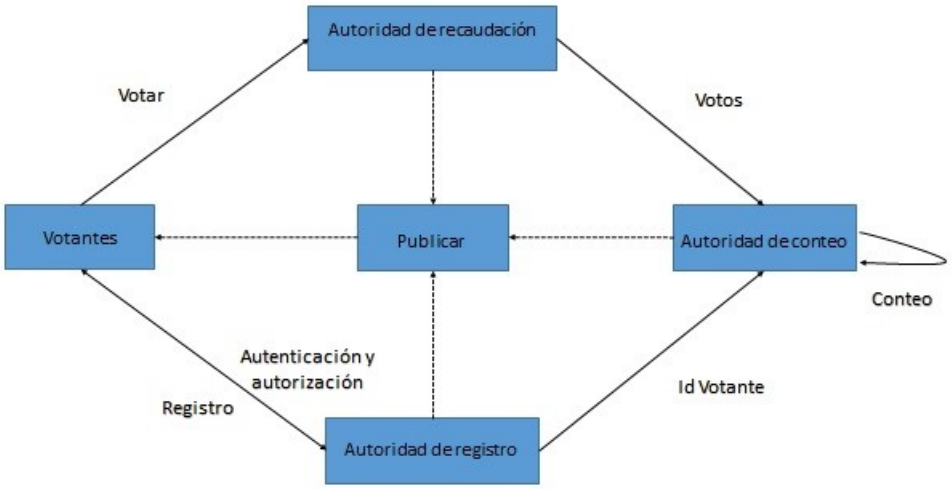

Imagen 6.1: Proceso de votación electrónica.

Ahora veremos las etapas que forman parte de este proceso:

1. Registro: La primera fase consiste en que las personas que son aptas para emitir su voto se identifiquen. Este proceso asegura que no se adultere la votación existiendo varios votos de un mismo votante.

2. Autenticación y autorización: Después hay que realizar la elección de que votantes son aptos para votar y no han votado hasta el momento en el sistema electoral. Estos votantes a su vez están dando su consentimiento para que su voto forme parte del proceso.

3. Votar: Los votantes se encargan de emitir su voto.

Nota 6.1. En nuestro estudio solo existirán dos opciones de voto 0 o 1.

4. Recuento: Por último se lleva el conteo de los votos, y una vez obtenido el resultado se pública.

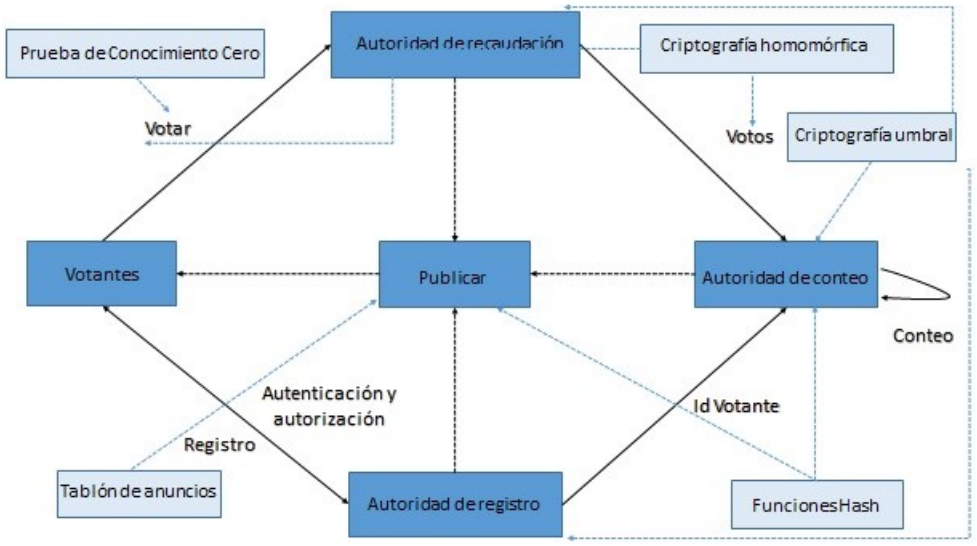

Imagen 6.2: Relación del proceso de votación electrónica y las herramientas criptográficas.

El tema de la seguridad es realmente importante en este protocolo, y existen diferencias entre si el voto se emite de manera electrónica o de manera tradicional. En la votación electrónica, la seguridad no se ve de manera visible. Los adversarios a los que el protocolo hace frente en numerosas ocasiones se encuentran en la etapa de emisión del voto. Debido a que esta fase se realiza por una red no controlada en su totalidad.

Por todo esto un buen sistema de votación electrónica tiene que cumplir una serie de objetivos que desarrollaremos a continuación:

Uno de los temas más importantes a tratar es la **privacidad** y durante todo el proceso es conveniente que no se produzca ninguna filtración de la información. La única información que se debe hacer pública es el resultado final. Además, el voto tiene que ser totalmente anónimo y no puede ser revelado ni durante ni después de la votación. Solo los votantes aptos pueden participar en la elección y no pueden emitir más de un voto.

Otra cuestión fundamental es la **robustez**. El proceso debe tener total seguridad ante todo tipo de ataques que intenten manipular los datos. Es necesario que el protocolo solo contabilice los votos que cumplen una validez total. Por último, el conteo de votos debe ser **verificable** por todo participante, para que el proceso sea totalmente limpio y transparente.

Dado que un sistema tiene que cumplir todos los requisitos de seguridad anteriores, la votación remota por Internet no es una tarea sencilla de implementar. Sin embargo, con la utilización de diversas herramientas criptográficas lo suficientemente potentes, podemos llegar a generar un protocolo totalmente seguro, sección 6.5. Por esto decidimos dedicar en esta documentación una sección que nos permitiera tener presente todas estas herramientas que permitirán una mayor eficiencia y seguridad en el proceso de votación electrónica, sección 3.2.

Es muy útil en este protocolo utilizar cifrados homomórficos, ya que nos otorgarán la capacidad de tener presente toda la combinación de datos sin necesidad de recuperar la información individual, lo que provoca mantener la confidencialidad de esta entidad. Esto también nos permite calcular datos cifrados sin la necesidad de tener que descifrarlos previamente.

El tablón de anuncios es el canal de transmisión público que puede ser utilizado por todas las partes que se involucran en el proceso. La información que se pública en dicho tablón es accesible a todos los participantes. Se tiene permisos de acceso o de lectura a este tablón, y por el contrario, no se tienen permisos para eliminar o adulterar alguna información ya publicada. El tablón de anuncios proporciona al proceso verificabilidad y precisión.

6.2. Protocolo de votación electrónica

En este punto explicaremos la votación electrónica que tiene como base el esquema $PVSS$, sección 4.2.1. Siempre que hablemos en este punto de protocolo nos referiremos a la votación electrónica, de lo contrario se especificará la clase de protocolo que estamos comentando.

El protocolo lo dividimos en tres partes debido a la complejidad del mismo. En la primera parte se describirá de forma sencilla, sin llevar a cabo ningún tipo de justificación ni prueba matemática. En la segunda parte veremos las justificaciones matemáticas, todas ellas coincidirán en orden con el primer punto. Finalmente mostraremos las pruebas en la tercera parte y sus posteriores ejemplos numéricos.

Una visión muy general del protocolo es permitir que los participantes puedan votar de manera conforme o desconforme a una consulta $(0$ o $1)$, y publicar la suma de dichos resultados en el tablón de anuncios. Cuando la votación haya finalizado, un grupo de participantes tomará los votos y calculará el resultado final. Si usamos MPC y los esquemas $PVSS$ ningún participante tendrá información de los votos emitidos por los demás participantes. Además, todos podrán validar la corrección y consistencia de la votación.

Una breve descripción del protocolo es dividirlo en tres partes. Una primera fase de incialización, una segunda de elección y por último el recuento de los votos. Detallaremos algo más estas fases.

1. Inicialización: Es la fase electoral anterior a la elección. Se publican los parámetros y requisitos del sistema y los protocolos de seguridad. Un usuario inicia sesión como administrador en el sistema, y elige los parámetros de seguridad y la cantidad de participantes que participarán en la elección. Por último, hay que comentar el cálculo de los parámetros del sistema en los que cada contador genera una clave pública y una privada. La clave pública se registra en el tablón de anuncios y cada votante firma sus credenciales.

2. Emisión de votos: En esta fase ya estamos en la elección real. Los votantes emiten el voto y se publican los resultados en el tablón de anuncios. Cada votante vota 1 o 0 y genera un secreto aleatorio. Además los participantes en la votación crean particiones del secreto para cada contador y lo cifran con la clave pública. Los participantes llegan a compartir una prueba llamada **DLEQ**, para demostrar la consistencia del secreto. Por último, cada votante proporciona evidencia de que su voto es válido, con la prueba $\mathbf{PROOF_U}$.

3. Recuento: Esto representa la fase que se produce después de la elección. El con-

teo de votos se realiza por las autoridades que se registraron en el proceso de inicialización. Estos suman los resultados y los publican en el tablón de anuncios. Al menos t participantes acumulan y descifran sus particiones. Por último se presenta el cómputo final del total de votos.

Para entender mejor el protocolo empezaremos explicando un ejemplo muy sencillo de votación electrónica, en la que todo el mundo colabora y se sumará todos los votos de los participantes:

Ejemplo 6.1. Si consideramos el esquema de Shamir umbral (t, n). Un participante j que es apto para votar, prepara su papeleta creando el vector de particiones $s_j : (y_j^1, \ldots, y_n^j)$ que guardará la información de su voto.

El participante j enviará el recurso compartido y_i^j al participante i.

Cada participante tiene su propio vector de particiones, el jugador j tiene $(y_j^1, \ldots, y_n^j)$.

El participante j calcula $\sum_{i=1}^{n} y_j^i = d_j$.

Entonces una vez calculado $(d_1, \ldots, d_n)$ se recupera el secreto asociado $\sum_{i=1}^{n} s_i$. Por último, el secreto es contado para dar el resultado final de la votación.

Ahora describimos las bases del protocolo dejando fuera tanto las pruebas como la justificación matemática que comentaremos más adelante. Para que sea más sencillo entender el protocolo limitaremos el cómputo de los votos a un número finito de contadores, de manera que tendremos m votantes y n contadores.

6.2.1. Inicialización

El tablón de anuncios se encarga de presentar todos los parámetros del sistema, y de la publicación de todos los elementos. Entre los parámetros que se utilizarán en el protocolo encontramos los generadores g y G, además del parámetro de seguridad t.

$$q \in R\{2^{l-1}, \ldots, 2^l\}, \ \textit{where} \ \ l > 1024$$

$$f \in R\{2, \ldots, 2q-1\} \longrightarrow g = f^2 \mod 2q \ +1$$

$$F \in R\{2, \ldots, 2q-1\} \longrightarrow G = F^2 \mod 2q+1$$

$$t \in \mathbb{Z}_q^* = \{1, 2, 3, \ldots, q-1\})$$

El elemento primo q, se elige uniformemente al azar de $\mathbb{Z}$. Sin embargo, en la práctica usamos el subconjunto 2^{l-1}, debido a los requisitos de seguridad. Los generadores g y G se calculan como cuadrados de los conjuntos f y F, la razón de por qué se realiza así lo explicaremos en el punto 6.3.2. Además, $2q+1$ también debe ser primo ya que trabajaremos con el esquema de compartición de secretos de Shamir.

El participante es el encargado de generar una clave privada x_i y una clave pública y_i.

Clave privada: $x_i \in_{\mathbb{R}} \mathbb{Z}_q^* = \{1, 2, 3, \ldots, q-1\}$

Clave pública: $y_i = G^{x_i}, \;\; i \in \{1, 2, 3, \ldots, n\}$

Al realizar el recuento se genera una clave privada elegida al azar de forma uniforme de $\mathbb{Z}_q^*$ donde q es primo. La clave pública se calcula en una exponenciación en G. Esto provoca que se de la seguridad que tiene el problema del logaritmo discreto.

6.2.2. Emisión de votos

Lo primero que realiza el votante es votar con la negación o afirmación correspondiente a 0 o 1 respectivamente. Después elige un secreto uniformemente aleatorio $s \in \mathbb{Z}_q$ y utiliza el esquema $PVSS$ para distribuir sus particiones. Esto provoca que se mantenga una combinación entre el secreto y el voto. Entonces se genera la construcción de un polinomio aleatorio de grado $t-1$, que evaluará las particiones de cada uno de los contadores.

El votante emite su voto: $v \in \{0, 1\}$.

Secreto aleatorio: $s \in_{\mathbb{R}} \mathbb{Z}_q$.

Polinomio aleatorio:$p(x) = s + \alpha_1 x^1 + \alpha_2 x_2 +, \ldots, +\alpha_{t-1} x^{t-1}, \;\; \alpha_j \in_{\mathbb{R}} \mathbb{Z}_q$.

Particiones secretas: $p(0) = s, p(1), p(2), \ldots, p(n)$.

El grado del polinomio viene precedido del grado que marque el parámetro de seguridad t. Después cada votante elige uniformemente al azar los coeficientes $\alpha \in \mathbb{Z}_q^*$ y un secreto aleatorio s. Utilizando el esquema de Shamir calculamos los recursos compartidos.

El votante es el encargado en repartir el recurso compartido cifrado y crea tanto la prueba $PROOF_U$ y $DLEQ$.

Se cifra el recurso compartido:

$$Y_i = y_i^{p(i)}, 1 \leq i \leq n$$

y llamamos **voto oculto** a:

$$U = G^{s+v}$$

el cual tiene el problema de DL que oculta el voto.

Todos los votantes crean particiones encriptadas para contar 1, contar 2, ..., contar n. Llamamos $p(i)$ a la partición en un punto correspondiente a un recuento dado. Las particiones se encuentran encriptadas usando la clave pública de cada participante y_i. El voto v es bastante simple ya que solo contiene los valores 0 o 1. Reutilizando el secreto s y agregándolo a el voto v, este se publica en el tablón de anuncios. Cuanto más grande sea s en U, nos encontramos ante un problema mucho más complejo.

También se publica en el tablón de anuncios las pruebas $PROOF_U$ y $DLEQ$. Mientras que $PROOF_U$ es la prueba que se encarga de verificar sin revelar ningún tipo de información que el voto es 0 o 1, la prueba $DLEQ$ muestra que las particiones están construidas de manera coherente y correcta.

6.2.3. Recuento de votos

Es necesario el conteo de los votos para su posterior resultado. Hay que utilizar las claves privadas para calcular colectivamente el resultado final de los boletos válidos. El recuento es capaz de descifrar particiones y publicar una prueba $DLEQ$.

Llamamos **multiplicador de recursos compartidos cifrados** a:

$$Y_i{}^* = (\prod_{j=1}^{m} = Y_{ij})(mod\ \ 2 \cdot q + 1)$$

La propiedad de compartir un secreto homomórfico nos asegura que cada recuento se encargue de multiplicar las particiones y descifrarlas. Sea Y_{ij} el valor Y_i calculado por el j-ésimo votante, que es la participación cifrada $Y_i = (y_i{}^{p(i)})$, siendo i el contador 1, el 2 y el 3, y así sucesivamente. La j el votante 1, el 2 y así sucesivamente. Por tanto, $Y_i{}^*$ es el multiplicador de particiones encriptadas para un recuento dado i. El participante i ahora puede descifrar el multiplicador $Y_i{}^*$ utilizando su clave privada x_i.

Llamamos multiplicador de particiones de descifrado de las particiones del participante i a:

$$S_i{}^* = (Y_i{}^*)^{\frac{1}{x_i}} (mod\ \ 2 \cdot q + 1).$$

Como este es el multiplicador de particiones, entonces no se revela ninguna información de la acción individual y se puede publicar de forma segura el resultado descifrado en el tablón de anuncios. Tenemos que saber que los contadores descifrarán las particiones, y publicarán una prueba $DLEQ$. Esto indicará que el descifrado se realizó correctamente.

Es necesario calcular el inverso de la clave x_i. Se puede usar el algoritmo de Euclides extendido, pero la autoridad maestra aplica la interpolación de Lagrange. Todo esto se produce después de que el participante haya publicado sus particiones descifradas S_i^*, y una autoridad maestra calcula la suma de los secretos de los votantes.

$$(S_1^*)^{\ell_1 \cdot (mod\ \ q)} \cdot (S_2^*)^{\ell_2 \cdot (mod\ \ q)} (S_n^*)^{\ell_n \cdot (mod\ \ q)} (mod\ \ 2 \cdot q + 1) = G^{\sum_{j=1}^{m} s_j}$$

Juntando todas las S_1^* podemos calcular la suma de los secretos, a partir de la fórmula de interpolación de Lagrange 4.1.3.2. Entonces multiplicamos S_1^*, que podemos reducirlo a la suma de los exponentes, que a su vez son iguales a la suma de los secretos. Una autoridad maestra calcula los votos y como último paso se aísla los votos para calcular el posterior resultado final.

Multiplicando U_j de los votantes obtenemos lo siguiente:

$$(\prod_{j=1}^{m} U_j)(mod\ \ 2 \cdot q + 1) = G^{\sum_{j=1}^{m} s_j + v_j}$$

y calculamos $G^{\sum_{j=1}^{m} s_j + v_j}$. Para proceder a aislar la suma de votos v en el exponente

podemos multiplicar $(\prod_{j=1}^{m} U_j)$ por la inversa de $(G^{\sum_{j=1}^{m} s_j+v_j})^{-1}$, de tal manera que:

$$G^{\sum_{j=1}^{m} s_j+v_j} \cdot (G^{\sum_{j=1}^{m} s_j})^{-1} = G^{\sum_{j=1}^{m} v_j}$$

Para resolver el cálculo de los votos procedemos a calcular $G^0, G^1, G^3, \ldots, G^{v_j}$ y el recuento final de votos será el exponente elevado en G. Para un mejor calculo de los votos también podemos proceder a utilizar un algoritmo más eficiente como el de Giant-steps Baby-steps.

6.3. Detalles del protocolo

Ahora comentaremos todas las justificaciones matemáticas del protocolo para detallar y comprender mejor el protocolo.

6.3.1. Inicialización

En el proceso de inicialización del protocolo se elige un número primo q, para construir $2q + 1$. El protocolo implica que hay que calcular un grupo de orden q, lo que implica esta propiedad que las operaciones en el exponente deben satisfacer $g^q = 1$. Si en el exponente realizamos la operación $mod\ \ q$, entonces tenemos $g^q = g^0$, y esta operación se puede realizar gracias a la compartición de secretos de Sharmir, el cual requiere un campo finito.

Ejemplo 6.2. Ahora veremos esto con un ejemplo numérico en el que tomando el generador $g = 2$ y el primo $q = 5$, entonces tenemos $2^5 \mod 5 = 32 \mod 5 = 2$.

Ahora tomamos el cuadrado de los números módulo primo, de tal forma que $2q + 1$, a lo que se denomina primo fuerte. En esta estructura matemática podemos elegir $b = a^2$, lo que implicaría que $b^q = 1 \mod 2q + 1$.

Usando los mismos valores que antes tenemos $(2^2)^5 \mod 11 = 1024 \mod 11 = 1$.

Utilizando el pequeño teorema de Fermat, Teorema 3.8, eligiendo q y b de forma que cumplan que $(a^2)^{q+1-1} \mod 2q + 1 = 1 = a^{2q} \mod 2q + 1 = 1$. Esto implica que en el exponente trabajamos con $(mod\ \ q)$ y en las bases con $(mod\ \ 2q + 1)$.

6.3.2. Elaboración de los generadores

La elección de los generadores se realiza aleatoreamente y su valor se encuentra entre 2 y $2q-1$. En este proceso se quita el 1, ya que si se eleva al cuadrado 1 es 1 y no se puede usar como un generador. Además, eliminamos también $2q$ porque $(2q)^2 = 1 \mod 2q+1$, lo que significa que el cuadrado de $2q$ también dará 1 y no se puede utilizar como generador.

6.3.3. Emisión de votos

Veremos cómo procedemos con el cálculo de Y_i, que es donde se encuentran los recursos compartidos cifrados con la clave pública de los contadores y_i.

$$\textbf{Votante 1} : Y_{1,1} = y_{1,1}^{p_1(1)}, Y_{2,1} = y_{2,1}^{p_2(2)}, .., Y_{n,1} = y_n^{p_n(n)}$$

$$\textbf{Votante 2} : Y_{1,2} = y_{1,2}^{p_1(1)}, Y_{2,2} = y_{2,2}^{p_2(2)}, .., Y_{n,2} = y_n^{p_n(n)}$$

$$\textbf{Votante 3} : Y_{1,m} = y_{1,m}^{p_1(1)}, Y_{2,m} = y_{2,m}^{p_2(2)}, .., Y_{n,m} = y_n^{p_n(n)}$$

Al proceder a calcular el cifrado de las particiones Y_i, cada votante calculará la anterior a cada uno de los contadores. Para aclarar esto, tenemos Y_{ij} donde i es el $i-$ésimo recuento y j es el $j-$ésimo votante refiriéndose a $Y_{1,1}, Y_{1,2}$, y así sucesivamente.

Ahora veremos cómo construir las variables C_j y X_i para las pruebas $DLEQ$ y $PROOF_U$. El votante publica las pruebas tanto de la emisión del voto como la distribución correcta del mismo. La prueba $PROOF_U$ usa la variable C_0 y la $DLEQ$ usa la variable X_i. A la hora de elaborar las pruebas cada votante se encarga crear las variables con **coeficientes ocultos**:

$$C_j = g^{\alpha_j}, j \in \{0, 1, 2, 3, \ldots, t-1\},$$

donde $\alpha_0 = s$, el **multiplicador de coeficientes ocultos** es:

$$X_i = \prod_{j=0}^{t-1} {C_j}^{ij} = g^{p(i)}, 1 \leq i \leq n$$

C_j contiene todos los coeficientes α_j, incluyendo el α_0 secreto, que se encuentran ocultos en el exponente de g, que se asegura por el problema DL.

Si tomamos $p(i)$ elevado en el exponente de g, tenemos como resultado $p(i) = \alpha_0 + \alpha_1 i^1 + \alpha_2 i^2 +, \ldots, +\alpha_{t-1} i^{t-1}$, el cuál es creado por cada votante en la fase de emisión de votos.

Teniendo en cuenta la siguiente declaración:

$$X_i = \prod_{j=0}^{t-1} {C_j}^{i^j} = \prod_{j=0}^{t-1} g^{(\alpha_j)^{i^j}} = g^{\sum_{j=0} \alpha_j \cdot i^j} = g^{\alpha_0 \cdot i^0 + \alpha_1 \cdot i^1 + \alpha_2 \cdot i^2, .., \alpha_{t-1} \cdot i^{t-1}} = g^{p(i)}$$

El votante tiene que calcular X_1, X_2 y X_3, lo que significa un X_i por cada una de las tres particiones que realiza. Además, en cada una de las particiones tiene que demostrar su honestidad.

$$X_1 = \prod_{j=0}^{t-1} {C_j}^{i^j} = C_0^{1^0} \cdot C_1^{1^1} \cdot C_2^{1^2} = g^{\alpha_0 \cdot 1^0 + \alpha_1 \cdot 1^1 + \alpha_2 \cdot 1^2} = g^{p(1)}$$

$$X_2 = \prod_{j=0}^{t-1} C_j^{i^j} = C_0^{2^0} \cdot C_1^{2^1} \cdot C_2^{2^2} = g^{\alpha_0 \cdot 2^0 + \alpha_1 \cdot 2^1 + \alpha_2 \cdot 2^2} = g^{p(2)}$$

$$X_3 = \prod_{j=0}^{t-1} C_j^{i^j} = C_0^{3^0} \cdot C_1^{3^1} \cdot C_2^{3^2} = g^{\alpha_0 \cdot 3^0 + \alpha_1 \cdot 3^1 + \alpha_2 \cdot 3^2} = g^{p(3)}$$

6.3.4. Recuento de votos

Calcularemos los valores Y_i^* y S_i^*. Mientras Y_i^* es el multiplicador de particiones cifradas para un recuento dado i y S_i^* es el multiplicador de descifrado de todas las particiones del participante i. Hallamos el **multiplicador de particiones cifradas**:

$$Y_i^* = (\prod_{j=1}^{m} Y_{ij}) (mod \ \ 2 \cdot q + 1) = y_i^{\sum_{j=1}^{m} p_j(i)}$$

El **multiplicador de particiones descifradas**:

$$S_i^* = (Y_i^*)^{\frac{1}{x_i}} (mod \ \ 2 \cdot q + 1) = G^{\sum_{j=1}^{m} p_j(i)}.$$

Entonces tenemos:

$$y_i^{\sum_{j=1}^{m} p_j(i)} = (G^{x_i})^{\sum_{j=1}^{m} p_j(i)} = G^{x_i \sum_{j=1}^{m} p_j(i)} = (G^{x_i \sum_{j=1}^{m} p_j(i)})^{\frac{1}{x_i}} = G^{\sum_{j=1}^{m} p_j(i)}$$

Derivando S_i^* aplicando la clave privada del contador Y_i^*. Debemos tener en cuenta que estamos llevando a cabo el proceso inverso multiplicativo de la clave privada en q. Los p_j son las evaluaciones por el j-ésimo votante, y se puede escribir como:

$$Y_1^* = y_1^{(p_1(1)+p_2(1)+p_3(1),..,p_n(1))} \qquad S_1^* = G^{(p_1(1)+p_2(1)+p_3(1),..,p_n(1))}$$

$$Y_2^* = y_2^{(p_1(2)+p_2(2)+p_3(2),..,p_n(2))} \qquad S_2^* = G^{(p_1(2)+p_2(2)+p_3(2),..,p_n(2))}$$

$$Y_n^* = y_n^{(p_1(n)+p_2(n)+p_3(n),..,p_n(n))} \qquad S_n^* = G^{(p_1(n)+p_2(n)+p_3(n),..,p_n(n))}$$

Cada contador puede publicar S_i^* e Y_i^*. Teniendo en cuenta que el exponente en y_i y G es la evaluación de cada votante en algún punto de un polinomio dado:

$$h(1) = p_1(1) + p_2(1) + p_3(1), .., p_n(1)$$
$$h(2) = p_1(2) + p_2(2) + p_3(2), .., p_n(2)$$
$$h(n) = p_1(n) + p_2(n) + p_3(n), .., p_n(n)$$

Publicación del recuento 1 : $S_1^* = G^{\sum_{j=1}^{m} p_j(1)} \qquad Y_i^* = y_i^{\sum_{j=1}^{m} p_j(1)}$

Publicación del recuento 2 : $S_2^* = G^{\sum_{j=1}^{m} p_j(2)} \qquad Y_i^* = y_i^{\sum_{j=1}^{m} p_j(2)}$

Publicación del recuento n : $S_n^* = G^{\sum_{j=1}^{m} p_j(n)} \qquad Y_i^* = y_i^{\sum_{j=1}^{m} p_j(n)}$

La elaboración del paso donde la autoridad maestra aplica la interpolación de Lagrange sería:

$$S_1^{*\ell_1 \cdot (mod\ \ q)} \cdot (S_2^*)^{\ell_2 (mod\ \ q)} \cdot (S_n^*)^{\ell_n (mod\ \ q)} (mod\ \ 2 \cdot q + 1) = G^{\sum_{j=1}^{m} s_j}$$

Podemos cambiar la S^* con G, y tenemos como resultado final en los exponentes la evaluación en algún polinomio en 0, el cuál equivale a la suma de secretos s calculado por los votantes.

$$= G^{\sum_{j=1}^{m} \ell_j p_j(1)} \cdot G^{\sum_{j=1}^{m} \ell_j p_j(2)} \cdot \cdots \cdot G^{\sum_{j=1}^{m} \ell_j p_j(n)}$$

$$= G^{\sum_{j=1}^{m} \ell_j p_j(1) + \sum_{j=1}^{m} \ell_j p_j(2) + \cdots + \sum_{j=1}^{m} \ell_j p_j(n)}$$

$$= G^{\sum_{j=1}^{m} (\ell_j p_j(1) + \ell_j p_j(2) +, \ldots, + \ell_j p_j(n))} = G^{\sum_{j=1}^{m} p_j(0)} = G^{\sum_{j=1}^{m} s_j}$$

El $\sum_{j=1}^{m} p_j(0)$ es la suma de los valores secretos de s para los votantes, y formalmente lo podemos ver como la evaluación de algún polinomio $h(0) = s_1 + s_2, \ldots, s_n$.

El paso donde la autoridad maestra calcula la suma de los secretos y los votos:

$$(\prod_{j=1}^{m} U_j)(mod\ \ 2 \cdot q + 1) = G^{\sum_{j=1}^{m} s_j + v_j}.$$

Al multiplicar U_j obtenemos la suma total tanto de de los secretos como de los votos en el exponente.

Los votantes calculan U:

$$\textbf{Votante 1} : U_1 = G^{s_1+v_1}$$

$$\textbf{Votante 2} : U_2 = G^{s_2+v_2}$$

$$\textbf{Votante n} : U_m = G^{s_m+v_m}$$

La autoridad maestra multiplica la U:

$$(\prod_{j=1}^{m} U_j) = U_1, U_2, \ldots, U_m = G^{s_1+s_2,\ldots,s_m+v_1+v_2,\ldots,v_m}$$

Para aislar los votos en el exponente multiplicamos $(\prod_{j=1}^{m} Uj)$ por el inverso de $(G^{\sum_{j=1}^{m} s_j})^{-1}$. Esto nos lleva a la justificación siguiente:

$$\frac{G^{\sum_{j=1}^{m} s_j + v_j}}{G^{\sum_{j=1}^{m} s_j}} = G^{\sum_{j=1}^{m} s_j + v_j - \sum_{j=1}^{m} s_j} = G^{\sum_{j=1}^{m} v_j}$$

6.4. Pruebas

Ahora toca tratar el punto de la justificación matemática de las pruebas $DLEQ$ y $PROOF_U$.

Empezaremos aportando la justificación para la prueba $DLEQ$. Lo haremos de manera interactiva y no interactiva entre el votante y el verificador. Por último, también explicaremos esta prueba entre contadores.

6.4.1. Prueba interactiva DLEQ entre votantes y verificador

La prueba $DLEQ$ significa la igualdad del logaritmo discreto. Se prueba que $X_i = g^{p(i)}$ y $Y_i = {y_i}^{p(i)}$, nunca dando a conocer $p(i)$. Si el probador es honesto tenemos $a_1 = g^w = g^r \cdot X_i^C$ y $a_2 = y_i^w = y_i^r \cdot Y_i^C$. El probador se encarga de calcular la misma cantidad de X_i como de particiones, esto significa que cada vez que se crea una partición, el probador tiene la obligación de proporcionar una prueba $DLEQ$.

Protocolo DLEQ

Tiene como **entrada**: g, X_i, y_i, Y_i donde $X_i = g^{\alpha_i}$ y $Y_i = y_i^{\alpha_i}$.

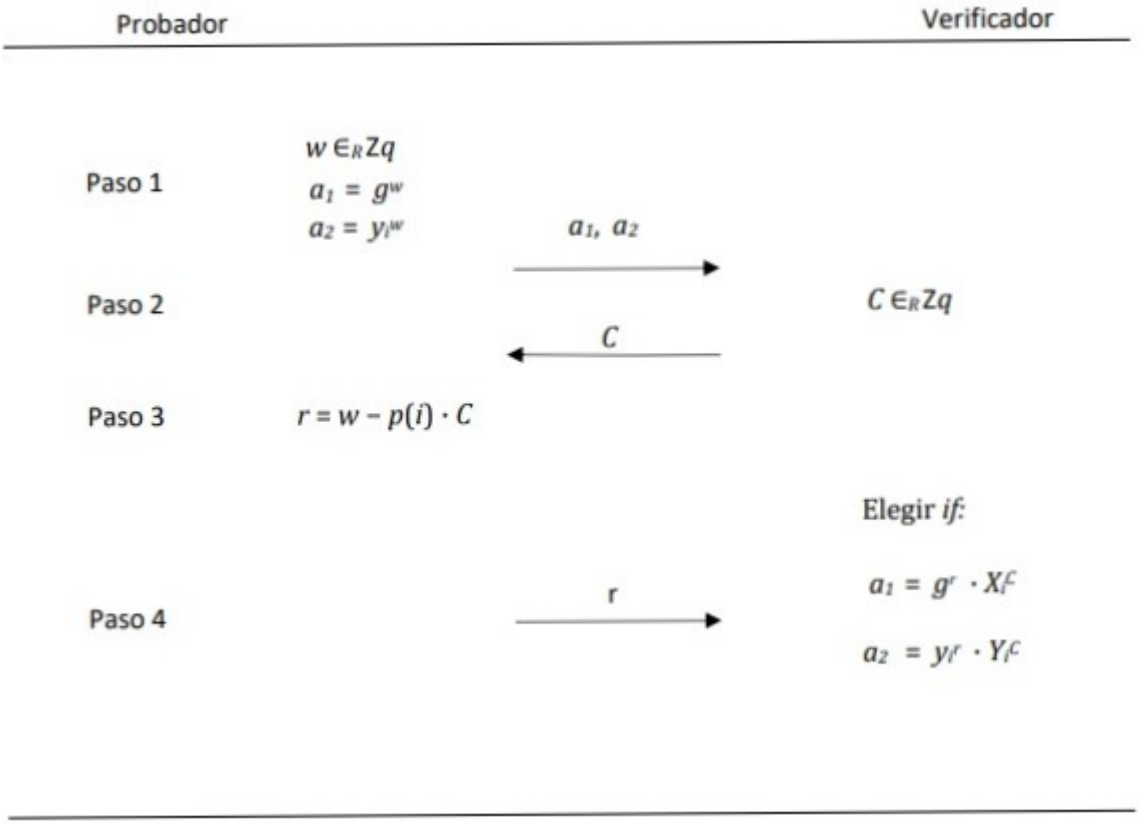

Imagen 6.3: DLEQ interactivo

El verificador envía un desafío C, después de que el probador ya se ha encargado de calcular a_1 y a_2. Solo se verifica el proceso si el probador usó los mismos exponentes que el verificador. La prueba dice que existe algún elemento α tal que $g^\alpha = X_i$ e $y_i^\alpha = Y_i$. Esta prueba DLEQ basa su funcionamiento en la prueba de conocimiento cero.

Correción DLEQ

La corrección se produce cuando todos los participantes son honestos, es decir, que tanto el probador como el verificador son honestos y la declaración está en lo cierto. Si esto ocurre la declaración siempre va a ser aceptada por el verificador.

La corrección se encarga de verificar $a_1 \stackrel{?}{=} g^w = g^r \cdot X_i^C$ y $a_2 = y_i^w \stackrel{?}{=} y_i^r \cdot Y_i^C$ están bien construidos. La corrección para a_1 es:

$$\begin{aligned} a_1 &= g^r \cdot X_i^C \\ &= g^r \cdot \left(g^{p(i)}\right)^C \\ &= g^r \cdot g^{p(i)\cdot C} \\ &= g^{r+p(i)\cdot C} \\ &= g^{w-p(i)\cdot C+p(i)\cdot C} \\ &= g^w \end{aligned}$$

La corrección de a_2:

$$\begin{aligned} a_2 &= y_i^r \cdot Y_i^C \\ &= y_i^r \cdot \left(y^{p(i)}\right)^C \\ &= y_i^r \cdot y^{p(i)\cdot C} \\ &= y_i^{r+p(i)\cdot C} \\ &= y_i^{w-p(i)\cdot C+p(i)\cdot C} \\ &= y_i^w \end{aligned}$$

Ahora veremos un ejemplo de esta prueba y el por qué de que sea necesario que el desafió sea aleatorio. Esto es necesario para dar solidez al proceso, ya que si el preparador le conoce puede preparar su partición y hacer trampa realizando una declaración incorrecta.

Ejemplo numérico DLEQ sin desafió aleatorio

1. El probador se encarga de enviar $a_1 = g^5, a_2 = y_i^6, X_i = g^3$ y $Y_i = y_i^4$ al verificador.

2. El verificador crea un desafío $C = 1$ para el probador.

3. El probador calcula $r = w - p(i) \cdot C = 5 - 3 \cdot 1 = 2$ y envía r al verificador.

4. El verificador tiene los siguientes datos $a_1 = g^2 \cdot X_i$ y $a_2 = y_i^2 \cdot Y_i$ y se encarga de verificar si:

$a_1 = g^2 \cdot X_i^C = g^2 \cdot g^{3 \cdot 1} = g^5$ y si: $a_2 = y_i^2 \cdot Y_i^C = y_i^2 \cdot y_i^{4 \cdot 1} = y_i^6$.

Vemos que los exponentes no coinciden, pero ambos controles son superados aunque el probador fuera deshonesto, por esto la importancia de que el desafío sea aleatorio. Ahora veremos un ejemplo en $DLEQ$ con un desafío aleatorio.

Ejemplo numérico DLEQ con desafió aleatorio

1. El probador se encarga de enviar $a_1 = g^5$, $a_2 = y_i^6$, $X_i = g^3$ y $Y_i = y_i^4$ al verificador.

2. El verificador crea un desafío $C = 9 \pmod 7$ para el probador.

3. El probador calcula $r = w - p(i) \cdot C = 5 - 3 \cdot 2 \pmod 5 = 4$ y envía r al verificador.

4. El verificador tiene los siguientes datos $a_1 = g^4 \cdot X_i^C$ y $a_2 = y_i^4 \cdot Y_i$ y se encarga de verificar:

$a_1 = g^4 \cdot X_i^C = g^4 \cdot g^{3 \cdot 2} = g^3 \cdot g^6 = g^9 = g^2$

$a_2 = y_i^4 \cdot Y_i^C = y_i^4 \cdot y_i^{4 \cdot 2} = y_i^4 \cdot y_i^8 = y_i^4 \cdot y_i^1 = y_i^5$.

En este caso el desafío aleatorio hace que la verificación no sea correcta, dado que $a_1 = g^5$ es diferente de $a_1 = g^2$, y $a_2 = y_i^6$ es diferente de $a_2 = y_i^5$, entonces la posibilidad de que falle gana probabilidad.

Justificación Matemática
Si el probador es deshonesto este fracasará ya que el verificador no sabe si el primer

paso $X_i = g^{p(i)}$ y $Y_i = y_i^{p(i)}$ se ha calculado correctamente. Si tomamos como exponentes $a_1 = g^w$ y $a_2 = y_i^{w'}$, además de $X_i = g^{m_i}$ y $Y_i = y_i^{m'_i}$. Sabemos que:

$$a_1 = g^w = g^r X_i^C = g^r \cdot g^{m_i C} = g^{r+m_i \cdot C}.$$

$$a_2 = y_i^{w'} = y_i^r \cdot Y_i^C = y_i^r \cdot y_i^{m'_i C} = y_i^r +^{m'_i} .^C.$$

y tenemos las desigualdades:

$$w = r + m_i \cdot C (mod\ \ q) \Rightarrow r = w - m_i \cdot C (mod\ \ q).$$

$$w' = r + m'_i \cdot C (mod\ \ q) \Rightarrow r = w' - m'_i\ \ C (mod\ \ q).$$

Estas dos desigualdades tienen que ser iguales, por lo que podemos reescribir:

$$w - m_i \cdot C = w' - m'_i (mod\ \ q) \Rightarrow (w - w') - (m_i - m'_i) \cdot C = 0\ \ (mod\ \ q)$$

Si la ecuación es cierta el probador no ha sido deshonesto, y como $m_i \neq m'_i$ la operación tendrá éxito. La C se conocerá y la construcción de w y w' tendrá como probabilidad $\frac{1}{q}$ para convencer al verificador.

Aclaremos esto con un ejemplo numérico. Teniendo en cuenta el caso de un probador deshonesto, si tomamos $C = 5$, $(w - w^i) - (m_i - m'_i) = 2$ y $q = 5$. El probador deshonesto no conseguirá su objetivo, pues $2 \cdot 5 = 0\ \ (mod\ \ 5)$. Cuanto más grande sea q, el probador deshonesto tendrá una gran probabilidad de fallar en la construcción.

Esta prueba se basa en el conocimiento cero, lo que significa que el verificador no aprende nada sobre $p(i)$. Para argumentar esto los valores enviados en el protocolo no dependen de $p(i)$. Entonces, se pueden construir a_1, a_2, r sin conocer $p(i)$, lo que quiere decir que no se puede aprender nada sobre $p(i)$.

6.4.2. Prueba no interactiva DLEQ entre votantes y verificador

En este punto explicaremos cómo podemos convertir la demostración interactiva del punto anterior 6.4.1, en una prueba no interactiva, para esto utilizaremos el protocolo de Fiat Shamir.

En este caso, al contrario que en el anterior, será el probador el que se encargue de calcular el desafío como si de una función aleatoria se tratase. Mostraremos dos

formas diferenciadas de realizar esta transformación, una versión optimizada y otra no optimizada.

Protocolo DLEQ no interactivo, no Optimizado

Tenemos como **entrada**: $g,\ \ X_i,\ \ y_i,\ \ Y_i$ donde $X_i = g^{x_i}$ e $Y_i = y_i^{x_i}$.

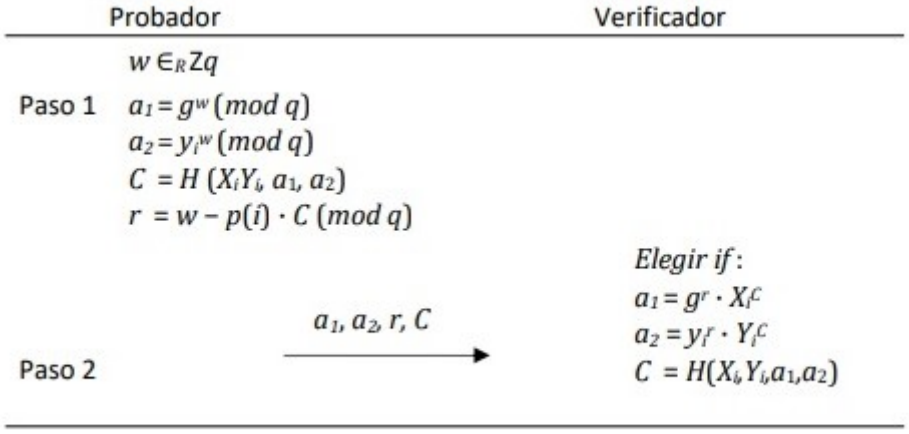

Imagen 6.4: DLEQ no interactivo

Protocolo DLEQ no interactivo, Optimizado

Mostraremos cómo se puede mejorar el proceso anterior de la figura 6.4 optimizando el cálculo del desafío C. Para esto calcularemos el desafío una vez y lo reutilizaremos, en vez de calcularlo n veces.

Paso 1. El probador publica: $a_1 = g^{w_i} \pmod q$ y $a_2 = y_i^{w_i} \pmod q$, $w \in_{\mathbb{R}} \mathbb{Z}_q$, $1 \leq i \leq n$. Además, calcula la función hash $C = H(X_i, Y_i, \ldots, X_n, Y_n, a_{1,1}, a_{2,1})$ y $r_i = w_i - p(i) \cdot C \pmod q$ y pública tanto r_i, como C.

Paso 2. El verificador se encarga de comprobar :

a) Si $a_{1,i} = g^{r_i} \cdot X_i^C \pmod q$.

b) Si $a_{2,i} = y_i^{r_i} \cdot Y_i^C \pmod q$.

c) Si $C = H(X_i, Y_i, \ldots, X_n, Y_n, a_{1,1}, a_{2,1}, a_{1,2}, a_{2,2}, \ldots, a_{1,n}, a_{2,n})$.

Por lo tanto, el valor hash contiene toda la información para que el probador calcule la prueba. Al igual que ocurre en la prueba interactiva, esta prueba se basa en el conocimiento cero, lo que significa que el verificador no aprende nada sobre $p(i)$.

Ejemplo 6.3. Este último proceso mejora la eficiencia de los $p(i)$, ya que sí existen por ejemplo 3 particiones distintas que se encargan de realizar el cálculo anterior. Este cálculo se realiza una vez por cada partición, pero todas con el mismo valor hash. MDetallaremos a continuación este proceso:

Paso 1. El probador publica $a_{1,1}, a_{2,1}, a_{1,2}, a_{2,2}, a_{1,3}, a_{2,3}, C, r_1, r_2, r_3$ en el tablón de anuncios. Además, selecciona $w_1, w_2, w_3 \in_{\mathbb{R}} \mathbb{Z}_q$ y calcula $a_{1,i} = g_i^w \pmod q$, $a_{2,i} = y_i^{w_i} \pmod q$, el hash $C = H(X_{i,i}, \ldots, X_n, Y_n, a_{1,1}, a_{2,1}, a_{1,2}, a_{2,2}, \ldots, a_{1,n}, a_{2,n})$ y $r_i : r_i = w_i - p(i) \cdot C \pmod q$.

Paso 2. La verificación contiene el siguiente cálculo:

(a) El verificador verifica si: $a_{1,i} = g^{r,i} \cdot X_i^C \pmod q$.

(b) El verificador verifica si: $a_{2,i} = y_i^{r_i} \cdot Y_i^C \pmod q$.

(c) El verificador verifica si: $C = H(X_{i,i}, \ldots, X_n, Y_n, a_{1,1}, a_{2,1}, a_{1,2}, a_{2,2}, \ldots, a_{1,n}, a_{2,n})$.

6.4.3. Prueba DLEQ entre contadores

Los contadores se encargarán de realizar cálculos en cada una de sus particiones. Entonces cada recuento utiliza la prueba $DLEQ$ para demostrar que el descifrado de sus particiones se realiza correctamente.

Así se prueba que los exponentes son iguales a $G = y_i^{x_i}$ y $Y_i = S_i^{x_i}$ sin revelar en ningún momento x_i y si el probador fue honesto. Se llega al caso en el que obtenemos que $a_1 = G^w = G^r \cdot y_i^C$ y $a_2 = S_i^w = S_i^r \cdot Y_i^C$.

En este caso realizaremos lo mismo que en la prueba entre el votante y el verificador. Mostraremos la prueba interactiva y la transformaremos de manera sencilla en una prueba no interactiva.

Protocolo DLEQ para contadores

Tenemos como **entrada** los valores: $G,\ y_i,\ \ S_i,\ Y_i$ donde $G = y_i^{x_i}$ e $Y_i = S_i^{x-i}$. Tenemos como valores iniciales que $g_1 = G,\ h_i = y_i,\ g_2 = S_i,\ h_2 = Y_i,\ \alpha = x_i$ y $w \in_R \{0, \ldots, q-1\}$.

Como **salida**: 0 o 1.

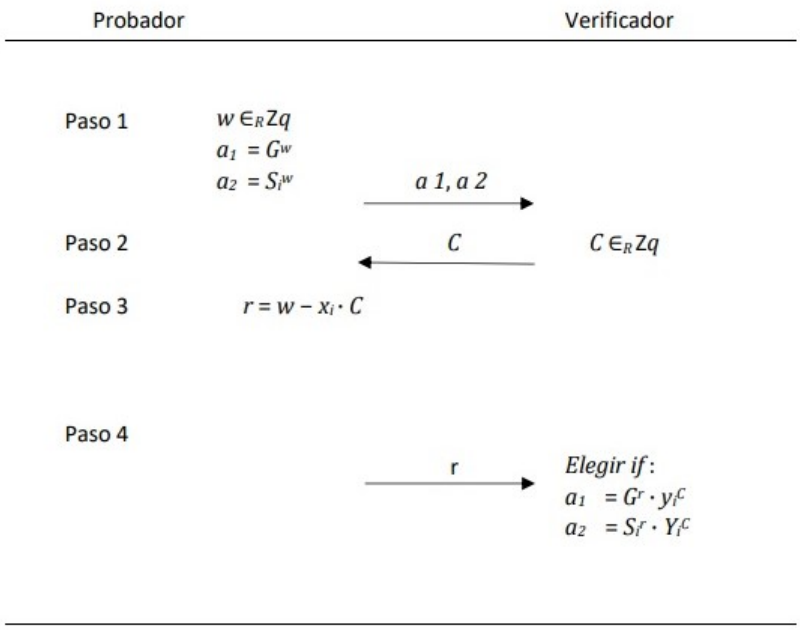

Imagen 6.5: Protocolo DLEQ para contadores

Nos fijamos en la figura 6.5, dado que en el **Paso 2** es donde el verificador crea el desafío para el probador, a partir de Fiat Shamir podemos transformar igual que antes una prueba interactiva en una prueba no interactiva. De esta manera remplazamos el paso dos por el valor hash $C = H(G, y_i, S_i, Y_i, a_1, a_2)$ tendríamos la prueba no interactiva.

Justificación Matemática

Podemos justificar la corrección de cálculos mostrada en el **Paso 4**, para ello podemos realizar la siguiente verificación en $a_1 = G^w \stackrel{?}{=} G^r \cdot y_i^C$ y $a_2 = S_i^w \stackrel{?}{=} S_i^r \cdot Y_i^C$.

$$a_1 = G^r \cdot y_i^C = G^r \cdot y_{x_i}^C = G^{r+x_i \cdot C} = G^{w-C \cdot x_i + x_i \cdot C} = G^w.$$

$a_2 = S_i^r \cdot Y_i^C = S^r \cdot Y_{x_i}^C = S^{w-C\cdot x_i} \cdot S_i^{x_i \cdot C} = S^{w-C\cdot x_i + x_i \cdot C} = S_i^w$.

Al igual que en las demás pruebas el verificador no tiene conocimiento de ninguna información de los $p(i)$.

6.4.4. Descripción de $PROOF_U$

Demostraremos la prueba $PROOF_U$ en la que el votante vota 1 o 0. Esta prueba se basará en dar coherencia a los exponentes, construyendo $U = G^{s+v}$ y $C_0 = g^s$. Los exponentes varían dependiendo de si el voto es 0 o 1.

Al igual que los casos anteriores transformaremos la prueba interactiva a través de Fiat–Shamir en una prueba no interactiva. Si el votante vota 0 entonces tendremos los pasos $1_a, 2, 3_a$ y 4 que observaremos a continuación. En cambio, si el votante vota 1, se seguirán los pasos $1_b, 2, 3_b$ y 4.

La diferencia entre votar 0 o 1 es simplemente intercambiar los valores entre las variables a_0, b_0 y a_1, b_1. Es importante saber que el verificador no podrá distinguir en ningún momento si el voto es 0 o 1.

Protocolo $PROOF_U$

Tiene como **entrada**: $U = G^{s+v}, C_0 = g^s$.

Paso 1: El votante vota 0 o 1:

a) El votante vota 0 y crea: $v = 0$, $w \in_{\mathbb{R}} \{1, \ldots, q-1\}$, $r_1 \in_{\mathbb{R}} \{1, \ldots, q-1\}$, $d_1 \in_{\mathbb{R}} \{1, \ldots, q-1\}$. Además, se encarga de publicar: $a_0 = g^w$, $b_0 = G^w$, $a_1 = g^{r_1} \cdot C_0^{d_1}$, $b_1 = G^{r_1} \cdot (\frac{U}{G^{1-v}})^{d_1} = G^{r_1} \cdot (\frac{U}{G})^{d_1}$.

b) El votante vota 1 y crea: $v = 1$, $w \in_{\mathbb{R}} \{1, \ldots, q-1\}$, $r_0 \in_{\mathbb{R}} \{1, \ldots, q-1\}$, $d_0 \in_{\mathbb{R}} \{1, \ldots, q-1\}$. Además se encarga de publicar: $a_1 = g^w$, $b_1 = G^w, a_0 = g^{r_0} \cdot C_0^{d_0}$, $b_0 = G^{r_0} \cdot (\frac{U}{G^{1-v}})^{d_0} = G^{r_0} \cdot (\frac{U}{1})^{d_0} = G^{r_0} \cdot U^{d_0}$.

Paso 2: El verificador crea el desafío $C \in_{\mathbb{R}} \{0, \ldots, q-1\}$ al votante.

Paso 3: Ahora el votante publica d_0, r_0, d_1, r_1. El votante vota 0 o 1:

a) Si el votante vota 0 se calcula: $v = 0$, $d_0 = C - d_1 \mod q$, $r_0 = w - s \cdot d_0 \mod q$.

b) Si el votante vota 1 se calcula: $v = 1,\ d_1 = C - d_0 \mod q,\ r_1 = w - s \cdot d_1 \mod q$.

Paso 4:En este paso el verificador calcula la coherencia, de tal manera que:

$$C = d_1 + d_0,\ a_0 = g^{r_0} \cdot C_0^{d_1},\ b_0 = G^{r_0} \cdot U^{d_0},\ a_1 = g^{r_1} \cdot C_0^{d_1},\ b_1 = G^{r_1} \cdot (\frac{U}{G})^{d_1}$$

Esto se pude convertir en una prueba no interactiva reemplazando el **Paso 2** por una función hash y evitando la participación con el verificador. Para esto se calculará un valor hash $C = H(U, C_0, a_0, b_0, a_1, b_1)$.

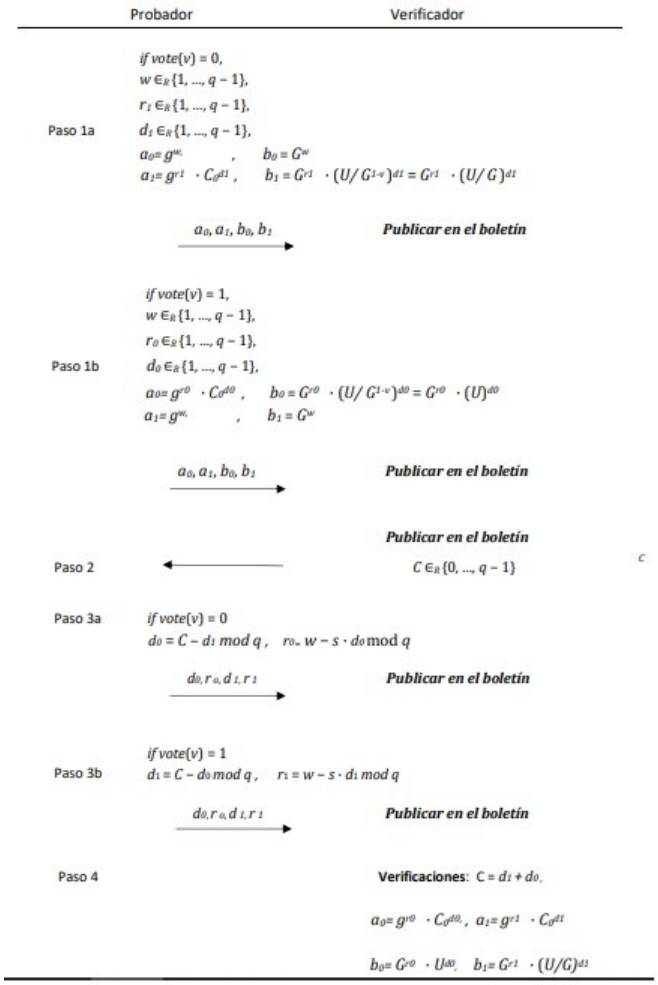

Imagen 6.6: $PROOF_U$

Justificación Matemática
La corrección se realiza derivando la justificación del **Paso 4**. Esto reemplazará a la expresión anterior de arriba por el valor real de la votación. Por lo tanto, tiene gran importancia el valor del voto.

Sabemos que $\mathbf{a_0} = \mathbf{g^{r_0}} \cdot \mathbf{C_0^{d_0}}$. Si el votante vota 0, $a_0 = g^w \stackrel{?}{=} g^{r_0} \cdot C_0^{d_0}$ está bien construido:

$$\begin{aligned} a_0 &= g^{r_0} \cdot C_0^{d_0} \\ &= g^{w-sd_0} \cdot g^{sd_0} \\ &= g^{w-sd_0+sd_0} \\ &= g^w \end{aligned}$$

Si el votante por el contrario vota 1, a_0 se construye a partir de $a_0 = g^{r_0} \cdot C_0^{d_0}$

Además, $\mathbf{b_0} = \mathbf{G^{r_0}} \cdot \mathbf{U^{d_0}}$. Si el votante vota 0, $b_0 = G_w \stackrel{?}{=} G^{r_0} \cdot U^{d_0}$ está bien construido:

$$\begin{aligned} b_0 &= G^{r_0} \cdot U^{d_0} \\ &= G^{w-sd_0} \cdot G^{(s+0)\cdot d_0} \\ &= G^{w-sd_0} \cdot G^{sd_0} \\ &= G^w \end{aligned}$$

Si el votante por el contrario vota 1, $b_0 = G^{r_0} \cdot (\frac{U}{G^{1-1}})^{d_0} \stackrel{?}{=} G^{r_0} \cdot U^{d_0}$ está también bien construido.

$$\begin{aligned} b_0 &= G^{r_0} \cdot (\frac{U}{G^{1-v}})^{d_0} \\ &= G^{r_0} \cdot (\frac{U}{G^0})^{d_0} \\ &= G^{r_0} \cdot (\frac{U}{1})^{d_0} \\ &= G^{r_0} \cdot U^{d_0} \end{aligned}$$

Sabemos que $\mathbf{a_1} = \mathbf{g^{r_1}} \cdot \mathbf{C_0^{d_1}}$. Si el votante vota 1, $a_1 = g^w \stackrel{?}{=} g^{r_1} \cdot C_0^{d_1}$ está bien construido:

$$\begin{aligned} a_1 &= g^{r_1} \cdot C_0^{d_1} \\ &= g^{w-sd_1} \cdot g^{sd_1} \\ &= g^{w-sd_1+sd_1} \\ &= g^{w} \end{aligned}$$

Si por el contrario vota 0, a_1 se construye a partir de $a_1 = g^{r_1} \cdot C_0^{d_1}$.

Además, $\mathbf{b_1} = \mathbf{G^{r_1}} \cdot (\frac{\mathbf{U}}{\mathbf{G}})^{\mathbf{d_1}}$. Si el votante vota 1, $b_1 = G^W \stackrel{?}{=} G^{r_1} \cdot (\frac{U}{G})^{d_1}$ está bien construido:

$$\begin{aligned} b_1 &= G^{r_1} \cdot (\frac{U}{G})^{d_1} \\ &= G^{w-sd_1} \cdot (U \cdot G^{-1})^{d_1} \\ &= G^{w-sd_1} \cdot (G^{s+1})^{d_0} \cdot G^{-d_1} \\ &= G^{w-sd_1} \cdot G^{sd_0+d_0} \cdot G^{d_1} \\ &= G^{w} \end{aligned}$$

Si por el contrario vota 0, $b_1 = G^{r_1} \cdot (\frac{U}{G})^{d_1} \stackrel{?}{=} G^{r_1} \cdot (\frac{U}{G^{1-v}})^{d_1}$ está también bien construido:

$$\begin{aligned} b_1 &= G^{r_1} \cdot (\frac{U}{G^{1-v}})^{d_1} \\ &= G^{r_1} \cdot (\frac{U}{G^{1-0}})^{d_1} \\ &= G^{r_1} \cdot (\frac{U}{G})^{d_1} \end{aligned}$$

6.4.5. Ejemplos numéricos del protocolo

6.4.5.1. Cálculos en el protocolo

A continuación se ilustrará un ejemplo numérico simple de los cálculos del protocolo, que mostrarán la emisión y el recuento final de los votos. Este ejemplo no

contienen cálculos de las pruebas, estas las hemos descrito en 6.4.

Tomamos 3 votantes (m) y 3 contadores (n). Enseñaremos cómo los 3 contadores pueden reconstruir la suma de todos los votos.

El **tablón de anuncios** es el encargado de publicar todos los parámetros del sistema: el número primo q, los generadores g y G y un parámetro de seguridad t.

La expresión ℓ se basa en la nota 4.5, desarrollada en el punto 4.1.3.2 y es calculada de la siguiente manera para 3 participantes estáticos:

$$\ell_1(0) = \prod_{j \in C, j \neq 1} \frac{2}{2-1} \cdot \frac{3}{3-1} = \frac{3}{1} = 3$$

$$\ell_2(0) = \prod_{j \in C, j \neq 2} \frac{1}{1-2} \cdot \frac{3}{3-2} = \frac{1}{-1}\frac{3}{1} = -3$$

$$\ell_3(0) = \prod_{j \in C, j \neq 3} \frac{1}{1-3} \cdot \frac{2}{2-3} = \frac{1}{-2}\frac{2}{-1} = 1$$

Parámetros	
Elemento primo q	5
Conjunto F	2
Generador G	4
Conjunto f	3
Generador g	9
Parámetros de seguridad t	3
ℓ_1, ℓ_2, ℓ_3	$3, -3, 1$

Tabla 6.1: Parámetros del protocolo.

El contador genera una clave privada x_i y una clave pública y_i.

Contadores		
	Clave y_i	Clave x_i
Conteo 1	4	1
Conteo 2	5	2
Conteo 3	9	3

Tabla 6.2: Claves públicas y privadas de los contadores.

Los votantes se encargan de emitir sus votos, ya sea 0 o 1. Además construyen el secreto aleatorio s y un polinomio aleatorio de grado como máximo $t-1$.

Votante 1	
Voto v	1
Secreto aleatorio s	2
Polinomio aleatorio $p(x)$	$2+2x+2x^2$
Votante 2	
Voto v	1
Secreto aleatorio s	4
Polinomio aleatorio $p(x)$	$4+3x+2x^2$
Votante 3	
Voto v	1
Secreto aleatorio s	6
Polinomio aleatorio $p(x)$	$6+x+2x^2$

Tabla 6.3: Voto, secreto y polinomio de cada votante.

Los votantes crean sus particiones p (x)				
Voto/punto	p(0)	Conteo 1	Conteo 2	Conteo 3
$p_1(x)=2+2x+2x^2$	2	3	3	2
$p_2(x)=4+3x+2x^2$	4	4	3	1
$p_3(x)=6+x+2x^2$	6	4	1	2

Tabla 6.4: Se calculan $p_1(1) = 3$ (mód 5), $p_1(2) = 3$ $(mod\ 5)$, $p_1(3) = 2$ $(mod\ 5)\ldots$

El votante distribuye el cifrado de las particiones.

Enciptación de las particiones $Y_i = y^{p(i)}$ usando los contadores de clave pública			
Voto/Contador	Conteo 1	Conteo 2	Conteo 3
Voto 1	$4^3 = 9$	$5^3 = 4$	$9^2 = 4$
Voto 2	$4^4 = 3$	$5^3 = 4$	$9^1 = 9$
Voto 3	$4^4 = 3$	$5^1 = 5$	$9^2 = 4$

Tabla 6.5: La encriptación consiste en aumentar la participación en el exponente de la clave pública de los contadores como $y_i^{p_j(i)} \ \ (mod\ \ 11)$.

El contador multiplica las particiones cifradas Y_i^* y se encarga de descifrar el multiplicador de particiones S_i^*.

Contador calcula Y^*	
Conteo 1	$Y_1^* = (9 \cdot 3 \cdot 3) = 81 = 4(mod\ \ 11)$
Conteo 2	$Y_2^* = (4 \cdot 4 \cdot 5) = 80 = 3(mod\ \ 11)$
Conteo 3	$Y_3^* = (4 \cdot 9 \cdot 4) = 144 = 1(mod\ \ 11)$

Tabla 6.6: El contador calcula Y^*.

Contador calcula S^*		
Conteo 1	$(x_1)^{-1} = 1 \cdot x = 1(mod\ \ 5) = 1$	$S_1^* = 4^1 = 4(mod\ \ 11)$
Conteo 2	$(x_2)^{-1} = 2 \cdot x = 1(mod\ \ 5) = 3$	$S_2^* = 3^3 = 5(mod\ \ 11)$
Conteo 3	$(x_3)^{-1} = 3 \cdot x = 1(mod\ \ 5) = 2$	$S_3^* = 1^2 = 1(mod\ \ 11)$

Tabla 6.7: El contador calcula S^*.

Una autoridad se encarga de aplicar la interpolación de Lagrange y de calcular los votos.

Aplicar ℓ a S^*	
$S_1^{*\cdot\ell_1}$	$4^{3(mod\ 5)} = 4^3(mod\ 11) = 9$
$S_2^{*\cdot\ell_2}$	$5^{-3(mod\ 5)} = 5^2(mod\ 11) = 3$
$S_3^{*\cdot\ell_3}$	$3^{1(mod\ 5)} = 1^1(mod\ 11) = 1$
Multiplicar S^* a la suma de los secretos	
$G^{\sum_{j=1}^{m} s_j} = S_1^{*\cdot\ell_1} \cdot S_2^{*\cdot\ell_2} \cdot S_3^{*\cdot\ell_3}$	$9 \cdot 3 \cdot 1(mod\ 11) = 5$

Tabla 6.8: Una autoridad maestra multiplica las particiones descifradas.

Los valores $U = G^{s+v}$	
U_1 para el votante 1	$4^{(2+1)} = 9(mod\ 11)$
U_2 para el votante 2	$4^{(4+1)} = 1(mod\ 11)$
U_3 para el votante 3	$4^{(6+1)mod\ 5} = 5(mod\ 11)$
Multiplicar U_i a la suma de los secretos y de votos	
$\prod_{j=1}^{m} U_j = G^{\sum_{j=1}^{m} s_j + v_j}$	$9 \cdot 1 \cdot 5(mod\ 11) = 1$

Tabla 6.9: Una autoridad maestra multiplica todos los votos.

Computar la suma final de votos $G^{\sum_{j=1}^{m} v_j}$	
Mostrar $U_i/(S_i^*)^{\ell} = G^{\sum_{j=1}^{m} s_j + v_j} / G^{\sum_{j=1}^{m} s_j}$	
$G^{\sum_{j=1}^{m} s_j + v_j}$	1
$G^{\sum_{j=1}^{m} s_j}$	5
Inversa de $(G^{\sum_{j=1}^{m} s_j})^{-1}$	$5 \cdot x = 1 (mod\ 11) = 5 \cdot 9 = 45 (mod\ 11) = 1$
$G^{\sum_{j=1}^{m} s_j + v_j} / G^{\sum_{j=1}^{m} s_j}$	$1 \cdot 9 (mod\ 11) = 9$

Tabla 6.10: Se puede aislar la suma de todos los votos multiplicando por el inverso.

Se generará una búsqueda exhaustiva para extraer el recuento final. En este caso el cálculo final consiste en resolver $G^x(\ \text{mód}\ 11) = \ 9$ donde x es el recuento total de votos. En este caso sería tres ya que en nuestro ejemplo teníamos tres votantes que votaron 1, por lo tanto, $4^3\ \ \text{mód}\ 11 = 9$.

6.4.5.2. Cálculo de DLEQ entre votante y verificador

Presentaremos un ejemplo simplificado de los cálculos a través de la prueba $DLEQ$, que describimos con anterioridad en el apartado 6.4.1. El ejemplo lo basaremos en los cálculos de la sección anterior. Usaremos el votante 1 con un polinomio $2 + 4x + 2x^2$ y el conteo $1, 2$ y 3 como verificadores. Esto implica que usaremos el cálculo de los valores $y_1 = 4,\ Y_1 = 9,\ y_2 = 5,\ Y_2 = 4,\ y_3 = 9,\ Y_3 = 4$ desde el $DLEQ$ necesita las variables $g,\ X_i,\ y_i,\ Y_i$. Como en este ejemplo tenemos 3 particiones, se realizarán las 3 pruebas $DLEQ$ correspondientes. Para calcular X_1, X_2, X_3 primero necesitamos calcular C_j.

El votante 1 **computa** $C_j(g^{\alpha})$		
	α_i	g^{α}
C_0	2	$9^2(mod\ 11) = 4$
C_1	4	$9^4(mod\ 11) = 5$
C_2	2	$9^2(mod\ 11) = 4$

Tabla 6.11: Los α son los coeficientes del polinomio del votante 1, donde $\alpha_0 = 2, \alpha_1 = 4, \alpha_2 = 2$.

El votante 1 **computa** $X_1 = \prod_{j=0}^{t-1} C_j^{1^j}$ **para compartir** $p(1)$		
	i^j	C^{i^j}
$C_0^{0^1}$	$1^0 = 1$	$4^1 = 4(mod\ 11)$
$C_1^{1^1}$	$1^1 = 1$	$5^1 = 5(mod\ 11)$
$C_2^{2^1}$	$1^2 = 1$	$4^1 = 4(mod\ 11)$
$X_1 = \prod_{j=0}^{t-1} C_j^{1^j} = g^{p(1)} = (4 \cdot 5 \cdot 4) = 3(mod\ 11)$		

Tabla 6.12: Cálculo de X_1 por el votante 1.

El votante 1 **computa** $X_1 = \prod_{j=0}^{t-1} C_j^{2^j}$ **para compartir** $p(2)$		
	i^j	C^{i^j}
$C_0^{0^2}$	$2^0 = 1$	$4^1 = 4(mod\ 11)$
$C_1^{1^2}$	$2^1 = 2$	$5^2 = 3(mod\ 11)$
$C_2^{2^2}$	$2^2 = 4$	$4^4 = 3(mod\ 11)$
$X_2 = \prod_{j=0}^{t-1} C_j^{2^j} = g^{p(2)} = (4 \cdot 3 \cdot 3) = 3(mod\ 11)$		

Tabla 6.13: Cálculo de X_2 por el votante 1.

El votante 1 **computa** $X_1 = \prod_{j=0}^{t-1} C_j^{3^j}$ **para compartir** $p(3)$		
	i^j	C^{i^j}
$C_0^{0^3}$	$3^0 = 1$	$4^1 = 4(mod\ 11)$
$C_1^{1^3}$	$3^1 = 3$	$5^3 = 4(mod\ 11)$
$C_2^{2^3}$	$3^2 = 9 = 4$	$4^4 = 3(mod\ 11)$
$X_3 = \prod_{j=0}^{t-1} C_j^{3^j} = g^{p(3)} = (4 \cdot 4 \cdot 3) = 4(mod\ 11)$		

Tabla 6.14: Cálculo de X_3 por el votante 1.

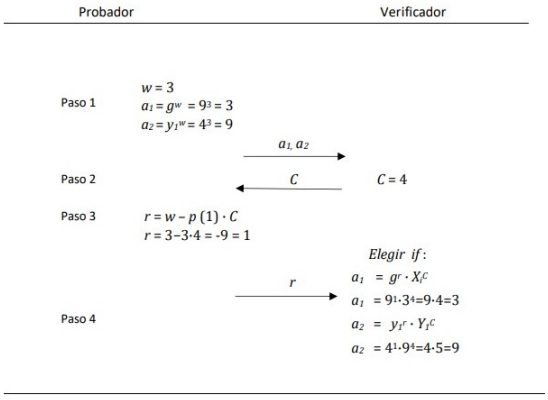

Imagen 6.7: Prueba DLEQ interactivo para X_1

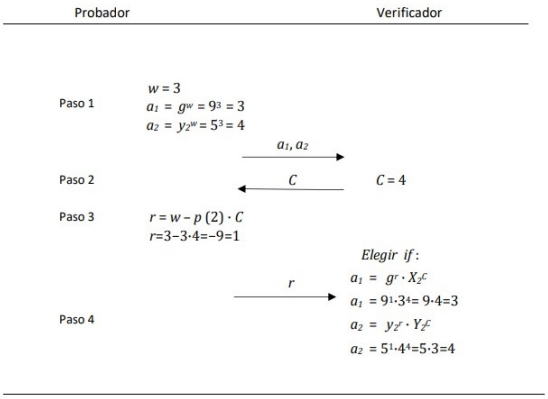

Imagen 6.8: Prueba DLEQ interactivo para X_2

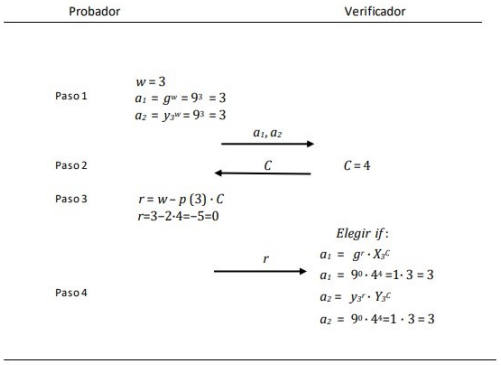

Imagen 6.9: Prueba DLEQ interactivo para X_3

6.4.5.3. Cálculo de $PROOF_U$ entre votante y verificador

Presentaremos un ejemplo simplificado de los cálculos a través de la demostración $PROOF_U$. Como se describe en la sección 6.4.4, se prueba que el voto es 0 o 1 sin revelar el valor real del voto. El ejemplo lo basaremos en los cálculos de la sección 4.3.5.1. El ejemplo mostrará un votante que vota 1. Entonces usamos el votante 1 con voto $v = 1$ y su secreto $s = 2$.

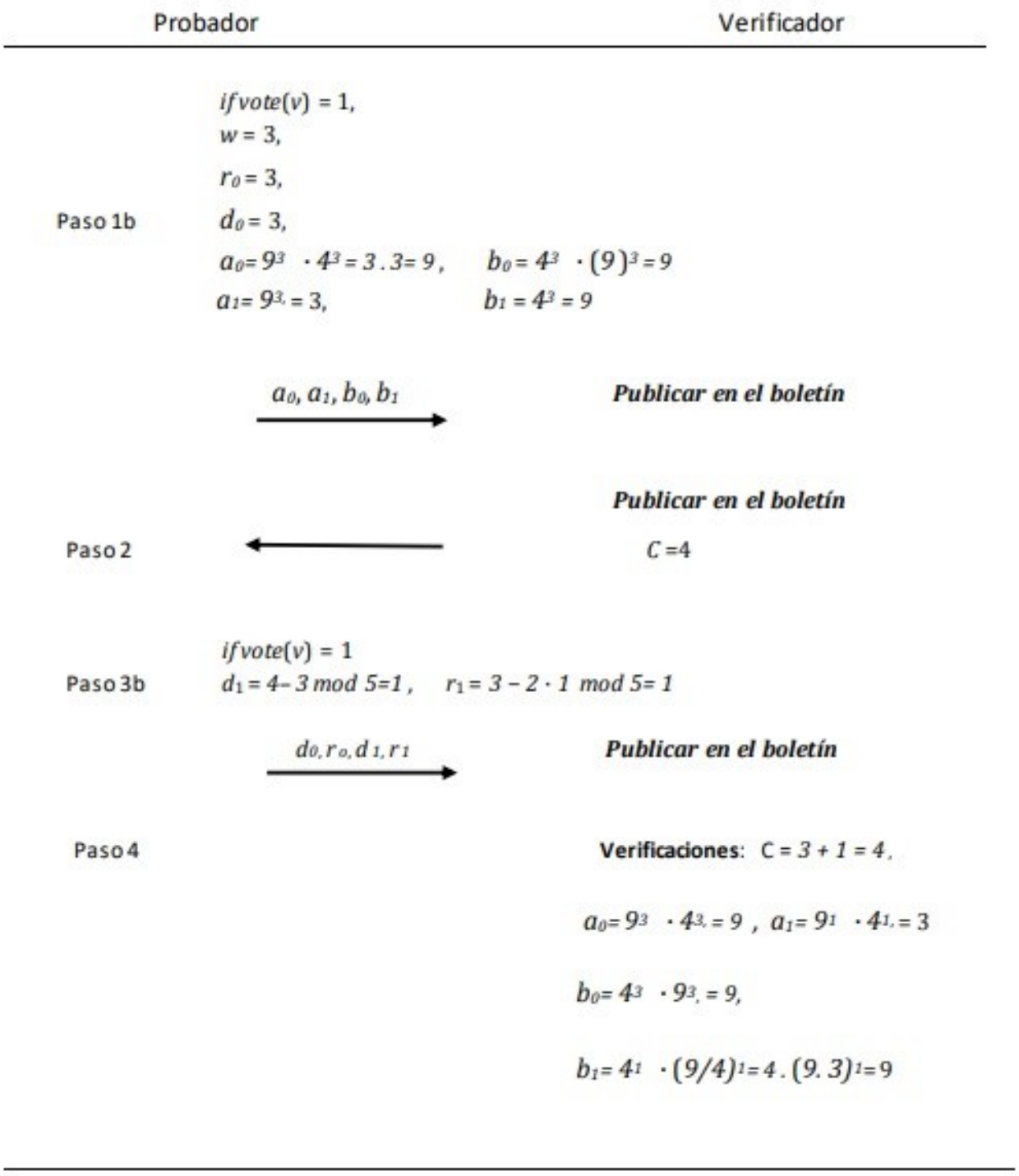

Imagen 6.10: Prueba $PROOF_U$

6.5. Seguridad

La seguridad significa poder defender toda la información y datos de accesos no autorizados, mientras se da acceso a las personas o sistemas autorizados a dicha información.

Siempre que hablemos de criptografía hay que dar la importancia que se merece a la seguridad. En el caso del protocolo de votación electrónica el concepto de seguridad tiene mucha importancia, pues desvelar cualquier tipo de información puede condicionar el protocolo.

En el proceso de mantener la seguridad del protocolo, cuando se detecta cualquier ataque, el sistema tiene que informar a todos los participantes. Si existen votos no válidos, el sistema debe marcarlos en la base de datos para que no se recuenten en el conteo final. Se intenta mostrar la coordinación y consistencia que existe entre la creación y verificación de las pruebas entre el probador, el tablón de anuncios y el verificador.

El voto es un proceso activo porque se espera constantemente las entradas de los usuarios para la verificación y el conteo. Después, el boleto es recibido por el tablón de anuncios y se procede al almacenamiento de las papeletas. El tablón de anuncios también es un proceso activo ya que siempre tiene abierto un puerto específico para poder mantener solicitudes entrantes de información. El recuento a su vez es también un proceso activo, ya que es el encargado de la verificación de las pruebas y marca en el tablón de anuncios un parámetro indicando si la prueba fue aceptada o rechazada. Del mismo modo, el almacenamiento es un proceso activo debido a que siempre espera la constante entrada de datos. El almacenamiento de boletos por el tablón de anuncios nunca actualizará los datos. Esta restricción permite al sistema que usuarios no autorizados no puedan cambiar cualquier dato del almacenamiento, tirando abajo todo el proceso.

6.5.1. Diseño de votación electrónica totalmente seguro

El diseño de votación electrónica que hemos explicado a lo largo de este capítulo es totalmente seguro. El protocolo $PVSS$ se encarga de asegurar que los votos son válidos, aceptados y contados, evitando que votos no autorizados sean contados y afecten el resultado. Para comprobar la integridad del mensaje se utilizan Funciones hash, además de las pruebas $DLEQ$ y $PROOF_U$.

Cuando el sistema detecta un ataque, este tiene que reaccionar contra dicho ataque. Con las pruebas $DLEQ$ y $PROOF_U$ se podrán detectar los ataques, además de la

verificación de la integridad del mensaje. Mientras que la prueba $DLEQ$ proporciona la coherencia en el proceso de cifrado, la prueba $PROOF_U$ asegura que el votante vote 0 o 1. Asimismo, la prueba $DLEQ$ garantiza que en la fase de recuento de votos, el descifrado se realice correctamente.

La manera de construir un secreto verificable públicamente con la utilización de los esquemas $PVSS$ como herramienta principal, nos permite obtener un esquema electoral simple y eficiente, en el que todos los participantes del esquema publican su mensaje. Existe un conjunto de contadores $A_1, \ldots, A_n$, un conjunto de votantes $V_1, \ldots, V_m$ y un conjunto de observadores pasivos. Un mismo participante puede ser tanto votante como contador.

Cualquier elección se desarrolla en dos fases, una primera fase en la que se emite el voto. Estos votos se publican de manera cifrada en el tablón de anuncios debido a que los votantes no necesitan ser anónimos. Además, hay que evitar el doble voto de un mismo participante. En la segunda fase, los contadores usan sus claves privadas para calcular el resultado final de la votación.

El proceso que se sigue en este protocolo es el siguiente; cada votante actúa en el esquema $PVSS$ como distribuidor, y los participantes juegan el papel de contadores. A la hora de iniciar el esquema $PVSS$, suponemos que cada A_i registra su clave pública y_i. Un votante V emite un voto $v \in \{0,1\}$ ejecutándose la distribución de protocolos para el esquema $PVSS$ descrita con anterioridad, utilizando como valor secreto aleatorio $s \in_R \mathbb{Z}_q$, y calculando el valor $U = G^{s+v}$.

Por tanto, el votante construye una prueba $PROOF_U$ que muestra sin revelar ninguna información que $v \in \{0,1\}$. $PROOF_U$ se refiere al valor de $C_0 = g^s$ que es publicado debido a el protocolo de distribución $PVSS$, y muestra que: $log_G\ U = log_g\ C_0 \bigvee log_G\ U = 1 + log_g\ C_0$

Podemos realizar una prueba de este tipo, tomando el voto del votante V, y el valor U y $PROOF_U$. Entonces la verificación pública del esquema $PVSS$ y de la prueba $PROOF_U$, hace que los boletos se puedan verificar en el tablón de anuncios, cuando los votantes envían dichos boletos.

En esta fase no se necesita la participación de los contadores ya que suponemos que los votantes $V_j, j = 1, \ldots, m$ han emitido votos válidos y el proceso de recuento de votos es ayudado por el protocolo de reconstrucción del esquema $PVSS$.

La primera acción a llevar a cabo es juntar todos los recursos compartidos cifrados,

es decir, calculamos los valores Y_i^* donde el índice j abarca todos los votantes:

$$Y_i^* = \prod_{j=1}^{m} Y_{ij} = y^{\sum_{j=1}^{m} p_j(i)}.$$

Ahora tomamos los A_i más altos y aplicamos el protocolo de reconstrucción al valor Y_i^*:

$$y^{\sum_{j=1}^{m} p_j(0)} = y^{\sum_{j=1}^{m} s_j}.$$

Aquí es donde entra en juego la propiedad homomórfica, y combinando $\prod_{j=1}^{m} U_j = G^{\sum_{j=1}^{m} s_j + v_j}$, obtenemos $G^{\sum_{j=1}^{m} v_j}$ y $T = \sum_{j=1}^{m} v_j$, $0 \leqslant T \leqslant m$, cuyo valor se puede calcular de manera eficiente.

6.5.2. Aplicación del protocolo diseñado.

Este esquema de elección tiene un gran nivel de seguridad, confiabilidad, privacidad y solidez. Además, no necesita un protocolo de generación de clave compartida para un descifrado de un esquema umbral. Estos protocolos de generación de claves necesitan varias rondas de interacción y deben ejecutarse entre niveles a gran escala, pues disminuye la complejidad para la propia elección.

En cambio, para elecciones pequeñas estos protocolos provocan un gran costo computacional. En elecciones a pequeña escala es más favorable eliminar esta generación de claves compartidas. Cuando el grupo de contadores varía de una elección a otra, entonces el esquema de elección no depende de la interacción entre los distintos votantes. Hay esquemas que necesitan un canal privado para interactuar con los votantes. Si cambiamos estos canales por la clave pública de cifrado, encontraríamos dos deficiencias principales:

- Las particiones de los contadores no son verificables públicamente, es decir, hay que crear una etapa en la elección en la que los participantes puedan quejarse. Estas quejas deben ser verificadas por los demás participantes. En el esquema $PVSS$ los contadores no participan durante la etapa de votación.

- El otro problema es que cada contador debe descifrar sus particiones una a una, esto implica que los recursos compartidos son comunicados. Además, se utiliza la clave para descifrar de manera individual, en lugar de descifrar solo una participación acumulada. Dado que nos conformamos con la privacidad computacional, concluimos que el esquema basado en $PVSS$ se puede utilizar para elecciones a menor escala.

El tema de la seguridad es la principal piedra que hace que estos sistemas no se instauren en la realidad. Pero una vez explicado un diseño seguro y un lanzamiento progresivo de este protocolo al mundo real, sería una solución para sistemas de votación que tienen muchos problemas con el censo, el registro y la elección transparente.

Siempre se debe tener en cuenta que crear una plataforma que esté lo suficientemente enfocada en la privacidad y la seguridad, requeriría una gran cantidad de gastos computacionales criptográficos. No obstante la importancia de este protocolo en la actualidad hace que matemáticos como Charles Hoskinson, integrante de empresas pioneras en el mundo del blockchain y monedas digitales, tengan equipos completos trabajando sobre el voto electrónico, intentando crear un proyecto consolidado y totalmente seguro.

Capítulo 7

Subastas electrónicas de oferta sellada

El crecimiento de las tecnologías en los últimos tiempos y la subida sin parar del comercio electrónico, han generado los protocolos de subastas electrónicas totalmente seguras. Este protocolo contiene una serie de participantes a los que llamaremos postores o licitadores, que envían ofertas selladas a los subastadores. El resultado del postor ganador se otorga sin revelar las ofertas perdedoras.

Las ofertas perdedoras no serán reveladas en nuestro protocolo, pues estos datos pueden ser usados en futuras negociaciones. Los subastadores pueden utilizar esta información para llegar a obtener datos acerca del estado del mercado y maximizar los ingresos, además de poder conseguir los productos deseados al menor precio posible.

Las subastas electrónicas de oferta sellada tienen como base los esquemas de compartición de secretos (ECS). Cada postor debe tomar la decisión de si está dispuesto a pagar el precio que se estipuló por el artículo. Para ello se elige un número entero distinto de cero que representa que "SÍ" se procederá a pagar ese precio. La elección de este valor es el precio que estaría dispuesto a pagar el postor por el artículo. Si un postor no está dispuesto a pagar un precio, elige cero para hacer saber que su decisión es "NO".

7.1. Fases de la subasta

La idea general de subasta es vender un artículo al precio más alto posible. Podemos definir las subastas en cuatro fases bien definidas:

1. Fase de preparación: Se publican las reglas que seguirá la subasta y se enseñan a los participantes.

2. Fase de licitación: En esta fase los postores presentan sus vectores de licitación sellados con el valor que ofrecerán por el artículo, por un canal de comunicación.

3. Fase de apertura de ofertas: En esta fase se produce la apertura de ofertas para determinar cual de ellas es la más alta.

4. Fase de determinación del ganador: Esta fase tiene como objetivo identificar el ganador, el cual consigue como obsequio la adquisición de un determinado artículo.

7.2. Propiedades de subasta de oferta sellada

Los protocolos de subasta selladas tienen una serie de propiedades que deben de cumplir:

- Corrección: el resultado de la subasta se determina de acuerdo con las normas estipuladas antes de la elección y de dictaminar el postor que resultó ganador.
- Confidencialidad de la oferta: cada oferta permanece confidencial para cualquier persona que no sea el propio participante que realizó la puja, antes de que comience la fase de la apertura de ofertas. Las ofertas perdedoras se mantienen en todo momento en secreto.
- Equidad: ningún postor puede aprovecharse de los demás postores. Esto implica no poder recuperar pujas, ni que exista la posibilidad de arrepentirse una vez que la oferta es realizada. Especialmente tener en cuenta que el ganador no puede negar su oferta una vez presentada.
- Verificabilidad pública: veracidad de la subasta, en este proceso se incluye la verificación pública de la validez de las ofertas. El proceso es completamente igual en la apertura de las ofertas y en la identificación del ganador.
- Privacidad de la oferta: La confidencialidad de las ofertas perdedoras se mantiene durante todo el proceso. Una vez terminado el proceso de subasta, esta información se mantiene en secreto. Solo se puede deducir la información que se revela del resultado final de la subasta.
- Solidez: la subasta se ejecuta correctamente en situaciones anormales, como si se produce una oferta no válida. Ninguna de las partes que participa en el protocolo es deshonesta.

7.3. Seguridad

En un esquema de subasta seguro, lo más importante es mantener en secreto el valor de las ofertas, es decir, la confidencialidad. No debe existir ninguna alianza de los licitadores con los subastadores, ya que esto supondría un desenlace fatal para la equidad de la subasta. Si esto ocurre los postores pueden conocer otras ofertas antes de enviar la suya, permitiendoles ganar la puja al precio más bajo posible.

El protocolo *MPC* juega un papel importante en la subasta electrónica ya que nos permitirá no revelar las ofertas de los licitadores, pues esta información puede decantar el protocolo para beneficiar a los participantes deshonestos.

La seguridad es bastante importante en este protocolo y se basa en el "Problema de los millonarios" [Yao]. Este problema es comentado en la sección MPC, ya que es el principio básico. En el protocolo cooperan n jugadores realizando una tarea de cálculo, basada en los datos privados que cada uno proporciona.

Ahora señalaremos distintos tipos de ataques que se pueden producir a las subastas de oferta sellada:

- **Ataque ABC** (subastador-postor): En este tipo de ataques el postor se alía con algunos subastadores, comprometiendo propiedades como la corrección o la equidad.

- **Ataque BBC** (postor-postor): En este tipo de ataques la alianza se produce entre los postores, comprometiendo la limpieza del protocolo.

- **Ataque de disputa**: En este tipo de ataques existe una disputa entre el subastador que acusa a un postor de presentar una oferta encriptada. El postor no puede probar su inocencia sin revelar la oferta.

La apertura de las ofertas depende del umbral que tenga el esquema de compartición de secretos. Si nos encontramos en el caso en el que hay varios subastadores deshonestos, las ofertas son abiertas ya que superan el umbral de compartición. Estas ofertas pueden ser reveladas a un postor que se encuentra a la espera de realizar su oferta. Esta acción provoca poder superar a las demás ofertas por un precio no muy superior, y terminar adquiriendo el artículo. Por lo tanto, el **ataque ABC** compromete claramente la equidad de la subasta debido a que los postores no están en igualdad de condiciones.

Si los protocolos de subasta de oferta sellada están basados en la compartición de secretos, el **ataque BBC** hace bastante vulnerables a estos tipos de esquemas. Los (ECS) condicionan poder conseguir la privacidad de las ofertas y la verificabilidad pública al mismo tiempo, ya que para conseguir la verificabilidad pública hay que privarse de la privacidad de la oferta.

Nos preguntaremos si se pueden prevenir todos estos ataques. La respuesta es sí, para ello realizaremos un diseño que prevendrá todo este tipo de ataques, sección 7.4.1.3. Este modelo tiene como base que el secreto verificable que se oculta computacionalmente pueda proteger la confidencialidad y privacidad de las ofertas.

7.4. Tipos de protocolo de subasta electrónica

Toda subasta tiene como resultado el licitador que presentó la oferta más alta para adquirir el artículo. A la hora de definir el precio que tiene la venta hay dos tipos de enfoque: **subastas de primer precio** (que es lo que se entiende a nivel general como subasta), en este tipo de subastas el postor que gana se encarga de pagar el precio que oferto, para adquirir el articulo. No obstante también existen **las subastas de segundo precio** (Vickrey) en las que el ganador paga el monto de la segunda oferta más alta. En las subastas de segundo precio se puede proponer un valor k en el que si el valor de la oferta es menor que k las posibilidades de llevarse el artículo disminuirán, y si el valor es mucho mayor que k puede que no sea una compra rentable para el licitador.

Tenemos que tener en cuenta que en las subastas de primer y segundo precio, un postor potencialmente puede llegar a definir al ganador y el precio de venta al mismo tiempo. Las subastas de primer precio son bastantes susceptibles a los ataques ABC. Por otro lado, en las subastas de segundo precio es bastante complicado lograr la privacidad de las ofertas, no existe un método práctico para lograr la privacidad en este tipo de subastas. Como nosotros primaremos la privacidad de dichas ofertas hablaremos en todo momento de subastas de primer precio a no ser que especifiquemos lo contrario. Mostraremos un diseño de subasta de primer precio de oferta sellada que previene todos los ataques descritos y que es totalmente eficiente y seguro.

7.4.1. Subasta electrónica basada en la compartición de secretos verificable

En el protocolo de subasta electrónica tiene bastante importancia el esquema de compartición de secretos verificable VSS, ya que estos nos permiten generar varios

tipos de ofertas selladas y que el proceso sea totalmente transparente.

Existen subastas de oferta sellada tanto de adversario pasivo como activo.
Los **adversarios pasivos** son participantes que siguen el protocolo al pie de la letra, pero tienen curiosidad por aprender información sobre las ofertas. Este modelo es seguro y no se genera mucha complejidad computacional.
En cambio, los **adversarios activos** son participantes que pueden no seguir el protocolo. Por lo tanto, en lo que a seguridad se refiere aplican costosos enfoques bit a bit. Cada oferta esta compuesta por varios bits y se utiliza un esquema VSS para cada bit de cada oferta, en lugar de utilizar un solo esquema VSS para todas las ofertas.

En la construcción inicial de los protocolos de subasta de oferta sellada de primer precio, se utilizan los esquemas VSS para dar un plus de seguridad al protocolo. Los subastadores tienen que abrir todas las ofertas para dar el resultado del ganador, por lo que conocen las ofertas perdedoras. Estas ofertas nunca son reveladas ni cuándo se completa la subasta. La idea para obtener los resultados de las ofertas es compararlas dígito por dígito, utilizando la compartición de secretos. Esto de comparar todos los dígitos es bastante costoso computacionalmete.

7.4.1.1. Introducción

En esta sección nos centraremos en dar un diseño de un protocolo de subasta electrónica de oferta sellada de primer precio. Comenzaremos explicando las subastas electrónicas que utilizan los esquemas de compartición de secretos homomórficos para sellar las ofertas. Este tipo de subastas unido a los esquemas de compartición de secretos VSS nos permitirán dar soluciones para tener un protocolo seguro a la vez que eficiente. Donde las ofertas perdedoras no serán sacadas a la luz en un modelo de adversario activo. Se dará seguridad sin tener una alta complejidad computacional utilizando la compartición de secretos.

También veremos dos tipos de subasta combinatoria segura:

(a) Subasta combinatoria por dinámica. Utilizaremos una programación dinámica existente para dar los resultados de la subasta. Este tipo de programación se basa en dividir un problema en subproblemas más pequeños y resolver cada subproblema hasta llegar a un caso base.

(b) Subasta combinatoria por TSP múltiple. Daremos un enfoque de negociación entre agentes para llegar a esclarecer los resultados de la subasta.

Describiremos modelos del que formarán parte n postores $B_1 \dots B_n$, m subastado-

res $A_1 \dots A_m$. Utilizaremos el esquema VSS para la distribución de la información, además de tener un participante de confianza al que llamaremos inicializador, cuya función es encargarse de distribuir la información. Aunque se pague un alto coste computacional se puede eliminar a este inicializador y sustituirlo por el protocolo MPC.

7.4.1.2. Subasta electrónica basada en el esquema de compartición de secretos homomórficos

Los protocolos de subastas de oferta sellada, aunque utilizan la compartición de secretos, siguen bastante expuestos a los ataques de participantes deshonestos. Ahora mostraremos un protocolo que otorgara total privacidad a las ofertas, en un umbral de confianza.

Los esquemas homomórficos son utilizados en la subasta electrónica por los postores para sellar sus ofertas. El esquema de Shamir y sus variantes, es el más utilizado por los postores para conservar la privacidad de la información. Se realiza una búsqueda binaria del precio ganador a lo largo de una ruta entre los precios que fueron propuestos. En la búsqueda, los subastadores para implementar la oferta abren todos los precios en la ruta de búsqueda binaria hasta que finalmente encuentran al ganador. Para realizar este proceso se utilizará los esquema VSS, dado que con este tipo de esquemas se aumenta la seguridad del protocolo.

Realizaremos un ejemplo de un ataque de disputa a una subasta de oferta sellada de primer precio que utiliza el esquema homomórfico. Este tipo de ataque de disputa solo se puede dar en subastas que no utilizan el esquema de compartición de secretos verificable. Este ejemplo nos ayudará a entender mejor la importancia de este tipo de protocolos y por qué usar el esquema VSS en nuestro posterior diseño.

Ejemplo 7.1. Vamos a suponer que tres pujadores B_1, B_2 y B_3 realizan el siguiente ataque contra una subasta sellada de primera oferta. Si tenemos un entero distinto de cero Y y el valor 0, que representan "SÍ" y "NO" respectivamente.

B_1, B_2 y B_3 intentan ganar la subasta superando en lo mínimo posible a la oferta más alta que denominaremos p_H. Para ello estiman que las ofertas de otros pujadores son inferiores a p_μ, mientras ellos ofertan p_ν superando el valor de p_μ.

Para llevar a cabo esta operación se reparten el trabajo. B_1 oferta Y a precios no superiores a p_μ y cero a otros precios; B_2 puja Y a precios no superiores a p_ν y cero a otros precios; B_3 puja $-Y$ a precios superiores p_μ pero no superiores a $p\nu$ y cero a otros precios.

En este caso si los postores hicieron ofertas inferiores a p_μ como se suponía, la suma de opciones en p_μ no es cero y la suma de opciones a precios superiores a p_μ es 0. Por lo tanto, p_μ sería el valor mayor y hay un empate entre B_1 y B_2, dado que los dos ofertaron Y. Entonces uno de los dos se rendiría y daría ganador al otro.

Ahora si la oferta más alta es p_H de otros postores, siendo esta no inferior a p_μ pero si menor que p_ν, la suma de opciones de p_H es mayor que cero y la suma de opciones a precios más alto que p_H es 0. En este caso ganaría el postor la subasta junto a B_2. Al mismo tiempo, B_2 todavía tiene la oportunidad de ganar la subasta en el desempate, para ello publica su oferta para ganar la subasta por el valor de p_ν.

Otra opción que se puede dar es que la oferta más alta es p_ν, y la suma de opciones en p_ν es mayor que cero. Además la suma de opciones a precios superiores a p_ν es 0. Entonces B_2 todavía puede ganar la subasta en el desempate.

Este ataque produce que B_1 o B_2 sean los ganadores de la subasta a no ser que un postor ajeno presente una oferta más alta que la que estimaron al principio (p_μ). Si las subastas están basadas en (ECS) no verificables entonces se puede dar el ataque disputa. Las disputas entre los postores y subastadores se producen al no poderse verificar las ofertas.

Así, para solucionar este ataque se suele utilizar con el esquema $PVSS$ pero esto puede producir otra serie de problemas, como llegar a revelar información sobre las ofertas. Para lidiar con este problema de confidencialidad y privacidad se utiliza el esquema de compromiso de Pedersen 4.4.1.

7.4.1.3. Ejemplo de subasta electrónica de primer precio seguro

El nuevo esquema del que hablaremos ahora muestra como un secreto verificable que se oculta computacionalmente puede proteger la confidencialidad y privacidad de las ofertas, siempre que el protocolo este bien diseñado. Para lograr esto se utilizarán los esquemas homomórficos. Los postores realizan pujas totalmente ocultas y se realiza una búsqueda binaria para dar el resultado del ganador de la subasta. El protocolo consta de dos rondas de comunicación entre los postores y los subastadores.

En la primera ronda los postores se encargan de comprometer las ofertas y de publicar el compromiso. Se oculta la información, de manera que no se puede recuperar ninguna de las ofertas. La función compromiso proporciona que no se puedan abrir los compromisos de varias maneras como en el problema del logaritmo discreto.
En la segunda ronda los postores se encargan de compartir la información para abrir las ofertas a los subastadores a través de un esquema VSS homomórfico aditivo. Entonces los subastadores cooperan entre ellos para recuperar la suma de las opciones

de licitación.

Fases del protocolo

Dado n postores $B_1, B_2, \ldots, B_n$ y m subastadores $A_1, A_2, \ldots, A_m$. Supongamos que hay w precios de oferta $p_1, p_2, \ldots, p_w$ en orden decreciente. Veamos las fases que se llevan acabo en el protocolo.

La fase de preparación, en esta fase se crea como canal de comunicación un tablón de anuncios entre los participantes para la transmisión de información. Entonces A_j establece su algoritmo de encriptación, cuya clave pública se llama N_j que se forma como producto de dos números primos grandes. El generador $g_j \in \mathbb{Z}_{N_j}$. La función de cifrado será $E_j(x) = g_j^x r^{N_j} \mod N_j^2$ y la de descifrado correspondiente $D_j()$.

A_j se encarga de publicar en el tablón de anuncios la función de cifrado unida a su clave pública para $j = 1, 2, \ldots, m$. Se eligen p y q que son números primos grandes tal que $p = 2q + 1$ y $nq^2 < N_j$ para $j = 1, 2, \ldots, m$. El grupo cíclico $G \in \mathbb{Z}_p^*$ con orden q. Se eligen números primos aleatorios y, g y h tal que $log_g\ y$ y $log_h\ g$ son desconocidos.

Por el esquema de compromiso de Pedersen sección 4.4.1, la función de compromiso de oferta es $Com(x, r) = y^x g^r \mod p$ donde x es una opción de oferta en $\mathbb{Z}_q$ y r es un entero de opción aleatoria en $\mathbb{Z}_q$. Elegimos el parámetro de compartición umbral T menor que m.

Todos los parámetros de los que hemos hablado p, q, y, g, h, T y N_j para $j = 1, 2, \ldots, m$, se encuentran publicados en el tablón de anuncios.

La fase de licitación, en la que cada licitador B_i se encarga de elegir el vector de licitación $(b_{i,1}, b_{i,2}, \ldots, b_{i,w})$ y sus opciones $p_1, p_2, \ldots, p_w$ donde $b_{i,l} \in \mathbb{Z}_q$ para $l = 1, 2, \ldots, w$. Si el postor se ve capacitado de pagar $p_l, b_{i,l}$, da el valor de un entero aleatorio distinto de cero módulo q; si no le interesa pagar nada sobre este artículo $p_l, b_{i,l} = 0$.

Después se firma y publica en el tablón de anuncios el compromiso $c_{i,l} = Com(b_{i,l}) = y^{b_{i,l}} g^{r_{i,l}} \mod p$ para $l = 1, 2, \ldots, w$ en el tablón de anuncios donde $r_{i,l}$ es elegido al azar de $\mathbb{Z}_q$. Consideramos hasta este paso la primera ronda de la subasta, en la que se comparte la opción de oferta $b_{i,l}$, pero no el precio p_l.

La fase de apertura de ofertas, una vez realizada las anteriores fases se configura la aleatorización de las ofertas. Los subastadores publican el compromiso, para esto pueden utilizar una función hash. Dado el entero aleatorio $R_{j,i,l} \in \mathbb{Z}_q$ para $i = 1, 2, \ldots, n$ y $l = 1, 2, \ldots, w$, cuando se publican todos los compromisos, los

subastadores se encargan de publicar $R_{j,i,l}$ para $j = 1, 2, \ldots, m$ como factores aleatorios de $b_{i,l}$ en el tablón de anuncios.

Después para compartir el secreto los propios participantes pueden verificar las particiones utilizando los esquemas VSS, como se explicó en la sección 4.4.1. B_i calcula $R_{i,l} = \sum_{j=1}^{m} R_{j,i,l} \mod q$. Calculamos $s_{i,l} = r_{i,l} \ R_{i,l} \mod q$ como el secreto en p_l para $l = 1, 2, \ldots, w$. B_i elige los polinomios $F_{i,l}(x) = \sum_{k=0}^{T} a_{i,l,k} \ x^k \mod q$ para $l = 1, 2, \ldots, w$, donde $a_{i,l,0} = s_{i,l}$ es el secreto que se desea compartir para $a_{i,l,k}$ con $k = 1, 2, \ldots, T$.

B_i publica el cifrado y comparte $S_{i,l,j} = E_j(F_{i,l}(j)) = g_j^{F_{i,l}(j)} t_{i,l,j}^{N_j} \mod N_j^2$ para $l = 1, 2, \ldots, w$ y $j = 1, 2, \ldots, m$ en el tablón de anuncios, donde $t_{i,l,j}$ se elige al azar de $\mathbb{Z}_{N_j}^*$. En esta segunda ronda de aleatorización se comparte $s_{i,l}$ para la apertura de las ofertas.

B_i publica compromisos de intercambio $C_{i,l,k} = h^{a_{i,l,k}} \mod p$ para $l = 1, 2, \ldots, w$ y $k = 0, 1, \ldots, T$ en el tablón de anuncios.

Los subastadores se encargan de realizar una búsqueda binaria. Si tenemos un precio p_l en la ruta de búsqueda, se realizan las siguientes operaciones:

Cada A_j calcula sus particiones sumadas: $v_{j,l} = D_j(\prod_{i=1}^{n} S_{i,l,j} \mod N_j^2)$.

Los compromisos correspondientes $u_{l,k} = \prod_{i=1}^{n} C_{i,l,k} \mod p$ para $k = 0, 1, \ldots, T$.

Luego se verifica que $h^{v_{j,l}} = \prod_{k=0}^{T} u_{l,k}^{j^k} \mod p$. En caso de que esto no se cumpla nos encontramos con que tenemos al menos una verificación fallida, entonces el postor B_I envía un recurso compartido de cifrado erróneo $S_{i,l,j}$. Entonces A_j publica en el tablón de anuncios $z_{i,l,j} = D(S_{i,l,j})$.

Entonces ahora pueden ocurrir dos cosas, que sea una falsa acusación por el subastador A_j y tengamos que eliminar A_j, o por el contrario que se tenga que eliminar al licitador B_i. Si se cumple que $h^{z_{i,l,j}} = \prod_{k=0}^{T} C_{i,l,k}^{j^k} \mod p \vee S_{i,l,j} \neq g_j^{z_{i,l,j}} t_{i,l,j}^{N_j} \ mod \ N_j^2$, A_j es eliminado, sino es así B_i es el que se elimina.

Ahora veremos como se recuperan los secretos homomóficos. Los A_j publican los $v_{j,l}$, la validez de ellos puede ser verificada por cualquier $C_{i,l,k}$ para $i = 1, 2, \ldots, n$ y $k = 0, 1, \ldots, T$. Si tenemos $T + 1$ particiones correctas sumadas, entonces el secreto se puede reconstruir (Esquema de Shamir):

$$d_l = \sum_{i=1}^{n} s_{i,l} = \sum_{j=1}^{T+1} v_{j,l}^{x_j} \text{ mód } q \text{ donde } x_j = \prod_{1 \leq k \leq t+1, k \neq j} \frac{k}{k-j} \text{ mód } q.$$

Para la apertura de los secretos homomórficos Si obtenemos como resultado $\prod_{i=1}^{n} c_{i_l}^{R_{i,l}} = g^{d_l}$ mód p, significa que la suma de de las opciones de ofertas aleatorias en p_l es cero, y la búsqueda binaria p_j termina negativamente y se disminuye dicha búsqueda hasta encontrar el ganador. Por el contrario si la suma de opciones de ofertas aleatorias de p_l no es cero, la búsqueda binaria p_j termina positivamente y se aumenta dicha búsqueda hasta encontrar el ganador.

La fase en la que se identifica al ganador. El resultado de toda subasta da como ganador a la mejor oferta que se realizó por uno o más artículos a subastar. Para identificar quién fue el ganador, supondremos que el precio del ganador es p_L, y las acciones descifradas $d_{i,L,j} = D_j(S_{i,L,j})$ para $i = 1, 2, \ldots, n$ se publican:

$$h^{d_i, L, j} = \prod_{k=0}^{T} C_{i,L,k}^{j^i} \text{ mód } p \text{ se verifica para } i = 1, 2, \ldots, n \text{ y } j = 1, 2, \ldots, m.$$

Si B_i consigue encontrar al menos $T + 1$ particiones correctas, su secreto $d_{i,L}$, puede ser recuperado por $d_{i,L} = \sum_{j=1}^{T+1} d_{i,L,j}^{x_j}$ mód p. Si se cumple que $C_{i,L}^{R_{i,L}} \neq g^{d_{i,L}}$ mód p solo existiría un único ganador que es B_i. No obstante B_i tiene que probar que él es realmente el ganador demostrando que conoce $log_y\ (C_{i,L}^{R_{i,L}} / g^{d_{i,L}})$ usando la prueba de conocimiento cero.

Una vez que es demostrado este último paso por B_i se publica su identidad, en caso que no lo realicé se elimina a B_i del proceso. Pueden existir más de un ganador, si esto se produce se realiza una nueva subasta entre los ganadores.

Seguridad del protocolo

Existen diversas diferencias entre los esquemas de subasta de primer precio que utilizan la compartición de secretos y los que utilizan los esquemas homomórficos basados en el intercambio de secretos existentes. En este esquema de subasta electrónica

a diferencia de las subastas basadas en la compartición de secretos, se prevendrán todos los ataques mencionados. Los puntos más fuertes del diseño que hemos descrito son evitar los ataques BBC y reforzar la privacidad de las ofertas, además de evitar los ataque de disputa.

El ataque ABC se previene gracias a que la función compromiso se encuentra escondida. Esta función también garantiza la inmutabilidad de la información, ya que esta no se puede modificar en ningún momento. El ataque BBC se evita gracias a que los subastadores suman las particiones de licitación aleatorias. Por último la utilización de los esquemas homomórficos eficientes permite resolver disputas de secretos compartidos. Además, al utilizar los esquemas VSS y al ser las opciones de licitación aleatorias, esto implica que no se revele ningún tipo de información de las ofertas perdedoras.

Con todos los ataques prevenidos el esquema de subasta que hemos diseñado es totalmente sólido y seguro. Todas las operaciones son publicadas y un postor honesto puede resolver cualquier disputa. Si los subastadores deshonestos no superan el umbral de participación, podemos llegar a conseguir la privacidad de las ofertas. Esto quiere decir que juntando varias herramientas criptográficas podemos resolver los ataques BBC, reforzando la privacidad de las ofertas sin desatender los ataques ABC y de disputa.

7.4.1.4. Protocolo de subasta combinatoria de oferta sellada por dinámica.

El protocolo de subasta combinatoria es seguro en un modelo de adversario pasivo. Para dar los resultados del ganador en este tipo de subasta utilizaremos la programación dinámica. El número de los subastadores es mayor que los máximos ingresos posibles que se pueden dar por los artículos que se encuentran ofertados en la subasta. Los licitadores envían la valoración de los grados de los polinomios, después los subastadores utilizan estos grados $deg(g_k) + deg(g_l) = deg(g_k \cdot g_l)$ y $max(deg(g_k), deg(g_l)) = deg(g_k + g_l)$, y por último se añadirá un cifrado homomórfico para que ninguna de las ofertas sea revelada.

Nos encargaremos de mostrar un ejemplo para que veamos como funciona el modelo de subasta combinatorio de oferta sellada por dinámica, después explicaremos el uso del método de programación dinámica. Para finalizar el protocolo se otorgarán los resultados y observaremos cómo todo esto funciona también en un entorno de adversario activo.

Ejemplo 7.2. Si tomamos seis postores $B_1, \ldots, B_6$ y todos proponen sus ofertas sobre tres artículos a, b, c. Si las ofertas son:

$\beta_1 = 2\$\ \{b\},\ \beta_2 = 3\$\ \{a\},\ \beta_3 = 2\$\ \{c\},\ \beta_4 = 4\$\ \{a,b\},\ \beta_5 = 8\$\ \{a,b,c\}$ y $\beta_6 = 10\$\ \{a,c\}$.

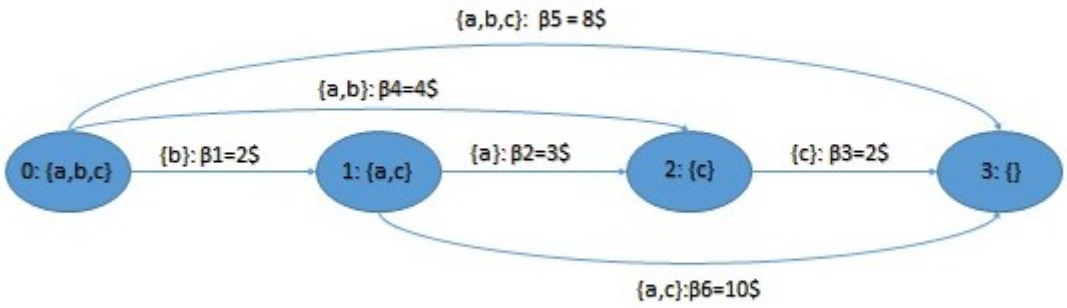

Imagen 7.1: Subasta combinatoria de oferta sellada por dinámica.

Entonces los subastadores obtendrán el mayor rendimiento si se encargan de vender los artículos $\{a,c\}$ a β_6 por 10\$ y el artículo $\{b\}$ a β_1 por 2\$.

De forma general tenemos $f(r) = \overset{r\longrightarrow s}{max}\ \{w_{rs} + f(s)\}$, donde w_{rs} es el peso del enlace entre dos nodos subsiguientes r y s. El valor del destino de la función es cero, es decir, utilizamos las operaciones de suma y máximo para llevar a cabo el protocolo de subasta combinatoria de oferta sellada, en un entorno de adversario activo.

Al igual que en la construcción anterior los licitadores $\beta_1, \ldots, \beta_n$ distribuyen las ofertas por polinomios simétricos de grado t. Y los subastadores $A_1, \ldots, A_m$ usan la operación suma para calcular $w_{rs} + f(s) = \beta_k + f(s)$. Por último, se aplica la comparación anterior para implementar la operación máxima, y poder definir los resultados finales de la subasta.

Tomando:

$r = 3 : f(3) = 0$ función de destino

$$r = 2 : f(2) = max\{w_{23} + f(3)\} = max\{1\} = 2$$

$$r = 1 : f(1) = max\{w_{12} + f(2), w_{13} + f(3)\} = max\{5, 10\} = 10$$

$$r = 0 : f(0) = max\{w_{01} + f(1), w_{02} + f(2), w_{03} + f(3)\} = max\{12, 6, 8\} = 12$$

Observamos como comentamos anteriormente que 12$ es el precio máximo que se puede obtener por la venta de la combinación de los productos, vendiendo por separado los artículos $\{a, c\}$ a β_6 por 10$ y el artículo $\{b\}$ a β_1 por 2$.

7.4.1.5. Protocolo de subasta combinatoria de oferta sellada por TSP múltiple

En este protocolo los vendedores cambian de lugar y vuelven a su lugar de origen después de completar el recorrido, por las diferentes combinaciones. Cada lugar es visitado una vez y el costo de la visita a los lugares se encuentra minimizado.

Ejemplificaremos el modelo de subasta combinatorio basado en el problema de los vendedores ambulantes múltiples, y demostraremos un enfoque de negociación entre agentes para resolver este problema.

Ejemplo 7.3. Si tenemos tres postores B_1, B_2, B_3, que generan ofertas a 7 artículos a, b, c, d, e, f, g de tal manera que:

$$B_1 \longrightarrow \{a, b, c\} : 14\$ \text{ o } \{a, b\} : 7\$ \text{ o } \{a, c\} : 9\$$$

$$B_2 \longrightarrow \{d, e\} : 7\$ \text{ o } \{b, d, e\} : 16\$ \text{ o } \{c, d, e\} : 12\$$$

$$B_3 \longrightarrow \{f, g\} : 8\$ \text{ o } \{b, f, g\} : 18\$ \text{ o } \{c, f, g\} : 14\$$$

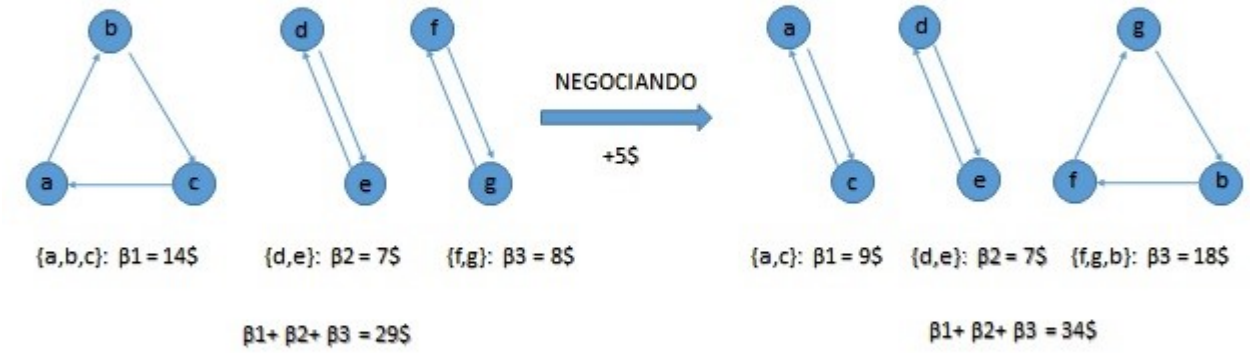

Imagen 7.2: Subasta combinatoria de oferta sellada por TSP múltiple.

En la primera fase los subastadores asignan todos los artículos a tres postores por el precio de 29$, y posteriormente en siguientes negociaciones intentan poder atribuirse el mayor de los beneficios de los productos. Los artículos $\{b\}$ y $\{c\}$ son liberados del primer licitador. Se realiza una oferta por $\{a, c\}$ que es mayor que la que se realizó por $\{a, b\}$. Entonces liberamos $\{b\}$ de tal manera que $14\$ - 9\$ = 5\$$. Lo mismo podemos realizar en B_2 $16\$ - 7\$ = 9\$ > 5\$$ por $\{b\}$ y B_3 paga $18\$ - 8\$ = 10\$ > 5\$$. Así, se venderá $\{b\}$ y el precio de venta se incrementará a 34$ negociando.

Se requieren de operaciones de adición y de comparación. El protocolo se implementa de forma segura y se limitan el número de negociaciones para llegar a dar una solución al protocolo.

Capítulo 8

Conclusiones

El proyecto es un estudio teórico de todos los conceptos y protocolos criptográficos necesarios para el desarrollo de la votación y subasta electrónica. Gracias al pseudocódigo de los algoritmos y la justificación matemática, nos ha permitido ejemplificar los resultados de manera práctico numérica. Todo este análisis ha implicado que prestemos una principal atención sobre la seguridad de la información.

El uso de dispositivos eléctricos en la votación y la subasta electrónica conlleva que el ámbito de la seguridad de la información y cómo preservarla sea la inquietud principal que nos ha surgido durante el proyecto. Este motivo ha implicado que tanto en la votación como en la subasta electrónica hayamos diseñado un proceso totalmente seguro.

Este proyecto nos ha permitido poner en práctica diferentes conocimientos vistos durante el Grado como puede ser la seguridad informática, criptografía y teoría de grupos. Sin embargo, la importancia fundamental del proyecto ha sido la adquisición de madurez y nuevas competencias, así como los objetivos prefijados en la sección 1.2, los cuales se han alcanzado en su totalidad.

1. El estudio exahustivo de las diferentes herramientas criptográficas nos permitió entender mejor las secciones posteriores, mientras que a su vez ha hecho posible profundizar en conceptos vistos durante el Grado como el intercambio de claves Difiie-Hellman o las Funciones hash.

2. El estudio de nuevas ramas criptográficas como los diferentes esquemas de compartición de secretos, nos permitirán preservar la información y mantenerla segura contra ataques externos.

3. El estudio de estos ataques externos de adversarios, ha hecho posible crear protocolos totalmente seguros con el uso de la computación segura entre múltiples

partes *MPC*, consiguiendo diseños sólidos y eficientes tanto de votación como de subasta electrónica.

4. El estudio general, justificación matemática y ejemplos numéricos realizados acerca del protocolo de votación electrónica han conllevado la adquisición de plenos conocimientos sobre este protocolo. Pensamos que sta estructura es una manera óptima de aprender el protocolo mientras se prueban los conceptos en la práctica.

5. El estudio de los diferentes tipos de subasta y del previo conocimiento sobre los esquemas de compartición de secretos nos ha permitido lograr todas las competencias sobre este protocolo, así como tener una percepción completa de todo el proceso, siempre garantizando la seguridad.

8.1. Ampliación de futuro

Para terminar el documento profundizaremos posibles mejoras que pueden contemplar un proyecto mucho más completo. Para ello diferenciaremos esta sección en los dos protocolos de peso en el documento: la subasta y la votación electrónica. Consideramos que el previo estudio teórico de las ramas criptográficas y de teoría de grupos es suficiente para que en futuras ampliaciones de personal como desarrolladores de software, puedan aprender los conceptos con una curva de aprendizaje menos pronunciada.

- **Votación electrónica** Como comentamos en la sección 6 solo se contemplan los votos $\{0, 1\}$, podríamos ampliar este concepto a votos nulos y a diversas elecciones. Además, los conocimientos previos de análisis y de diseño sobre el protocolo, suponen que podamos desarrollar además una aplicación que lleve a cabo el algoritmo y nos permita digitalizar el proceso. Esta aplicación de votación electrónica ha de ser escalable y aunque se tenga experiencia práctica en programación, nos podremos enfrentar a situaciones en las que la teoría nos ayudará a aclarar cómo resolver distintas situaciones. Por este motivo es totalmente beneficioso adquirir un conocimiento detallado de los conceptos matemáticos y conceptos de criptografía que hay detrás, no sólo del protocolo en sí mismo.

- **Subasta electrónica** Como comentamos en la sección 7, existen diferentes tipos de subasta y centramos nuestro estudio en las de primer precio, aunque también podríamos haber realizado un estudio de la misma forma de las subastas de segundo precio. Además, el análisis y diseño implementado con la adquisición plena de todos los conocimientos en este protocolo provocan que en futuras ampliaciones podamos meter el desarrollo de una aplicación que se encargue de

llevar a cabo todo este proceso.

Ambos diseños de las aplicaciones tanto de votación como de subasta electrónica han de estar influenciados por los requisitos de seguridad de los protocolos. Consideramos que estos requisitos son una parte esencial del desarrollo de una aplicación, por eso la importancia de los diseños seguros creados previos al desarrollo de la aplicación, secciones 6.5 y 7.4.1. Los requisitos se tienen en cuenta en ambos protocolos y se pretenderá cumplir atributos de calidad (QA) y patrones de diseño. Además, dotar a estos atributos de características fundamentales como escalabilidad, modificabilidad, seguridad y comprobabilidad sobre los que basar la arquitectura.

Bibliografía

[1] BLAKLEY, G. R. (1979). *Safeguarding cryptographic keys*. Proceedings AFIPS 1979 National Computer Conference, 48, 313-317.

[2] CASCUDO, I., *Secret Sharing Schemes with Algebraic Properties and Applications*, Department of Mathematics, Aalborg University, Aalborg, Denmark 2016.

[3] CETINKAYA, O., *Cryptography in electronic voting systems*, International Conference on Government and Governance, 2009.

[4] CETINKAYA, O., *Analysis of security requirements for cryptography voting protocols (extended abstract)*, Third International Conference on Availability, Reliability and Security, 2008.

[5] CHEN, H. R. CRAMER, R. DE HAAN, R., et al. *Strongly multiplicative ramp schemes from high degree rational points on curves*. N. Smart (Ed.), EUROCRYPT 2008, LNCS, vol. 4965, Springer, Heidelberg (2008), pp. 451 − 470.

[6] CHEN, H., CRAMER, R., GOLDWASSER, S., DE HAAN, R AND VAIKUNTANATHAN, V., *" Secure Computation from Random Error Correcting Codes,"* In Proceedings of 26th Annual IACR EUROCRYPT, Barcelona, Spain, Springer Verlag LNCS, vol. 4515, pp. 329-346, May 2007.

[7] DELGADO, F., AND NÚÑEZ, A, *Esquemas para compartir secretos*, Universidad de Valladolid 2018.

[8] DAMGÅRD, I. GROTH, J. AND ALOMONSEN, G., *The Theory and Implementation of an Electronic Voting System*, Kluwer Academic Publishers, 2002.

[9] GUTIÉRREZ, P., *¿Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales*, Genbeta , 2020, https://www.genbeta.com/desarrollo/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales.

[10] GONZÁLEZ M. I., *Criptografía Avanzada*, Universidad Rey Juan Carlos 2018.

[11] HAIFA, I., *Un curso intensivo sobre MPC*, Medium , 2020, https://medium.com/applied-mpc/a-crash-course-on-mpc-part-2-fe6f847640ae.

[12] HAO CHEN, RONALD CRAMER, SHAFI GOLDWASSER, ROBBERT DE HAAN AND VINOD VAIKUNTANATHAN, *Secure Computation from Random Error Correcting Codes*, Department of Computing and Information Technology, School of Information Science and Engineering, Fudan University, Shanghai, China 2007.

[13] JUSTESEN J. AND, HØHOLD, T., *A Course in Error-Correcting Codes*, European Mathematical Society Publishing House 2004.

[14] KATZ, J. AND LINDELL, Y., *Introduction to Modern Cryptography: Principles and Protocols*, Chapman y Hall, 2014.

[15] MALKHI, D., *An advance course in computer and network security. Lecture Notes*, 2002, Disponible en: http://www.cs.huji.ac.il/ns/SS.doc.

[16] MASSEY, J. L, *"Some applications of coding theory in cryptography,"* in P.G Farrell (ed.), Codes and Ciphers, Cryptography and Coding IV, Formara Lt, Esses, England, pp. 33-47, 1995.

[17] MCELIECE, R. AND SARWATE, D., *"On Sharing Secrets and Reed-Solomon Codes,"* Communications of the ACM, vol. 24, pp. 583-584, 1981.

[18] MEHRDAD NOJOUMIAN1 Y DOUGLAS R. STINSON, *Efficient Sealed-Bid Auction Protocols Using Verifiable Secret Sharing*, Department of Computer Science Southern Illinois University, Carbondale, Illinois, USA 2017.

[19] MEIJERING, E. (2002), «*A chronology of interpolation: from ancient astronomy to modern signal and image processing*», Proceedings of the IEEE 90 (3): 319-342.

[20] MUNUERA, C. AND TENA, J., *Codificación de la Información*, Universidad de Valladolid 1997.

[21] MUÑOZ I. , *Compartición de secretos y votación electrónica:implementación y aplicaciones*, Trabajo fin de grado matemáticas, 2021.

[22] (PADRÓ, C., *Lecture Notes in Secret Sharing*, Nanyang Technological University, Singapore 2013.

[23] PENG, KUN, COLIN, DAWSON EDWARD, *Optimization of Electronic First-Bid Sealed-Bid Auction Based on Homomorphic Secret Sharing*, Queensland University of Technology 2005.

[24] QI CHEN, DINGYI PEI, CHUNMING TANG, QIANG YUE, TONGKAI JIA. *Note on ramp secret sharing schemes from error-correcting codes*, 2011.

[25] RUANO, D., *Esquemas de compartición de secretos en rampa*, IMUVA, 2018.

[26] SCHOENMAKERS, B., *A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting*, Department of Mathematics and Computing Science Eindhoven University of Technology 1999.

[27] SHAMIR A, *A polynomial time algoritm for breaking the basic Merckle-Hellman cryptosystems*, IEEE trans on inform 1984.

[28] SHAMIR, A. (1979). *How to share a secret*. Communications of the ACM, 22, 612-613.

[29] THOMAS M. COVER, JOY A. THOMAS, *Elements of Information Theory* , City College of New York 2006.

[30] TRAPPE, W. *Introduction to Cryptography*, Prentice Hall, 2002.