INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE FÍSICA Y MATEMÁTICAS

VOTACIÓN ELECTRÓNICA

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO MATEMÁTICO

P R E S E N T A:

CHAVEZ LUNA CELIC AARON

ASESOR:

DR. LUIS CARLOS CORONADO GARCÍA

MÉXICO, D.F., 11 DE MARZO DE 2008

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE FÍSICA Y MATEMÁTICAS

Votación Electrónica

T E S I S

que, para obtener el título de

Ingeniero Matemático,

presenta

Chavez Luna Celic Aaron

ASESOR:
Dr. Luis Carlos Coronado García

México D. F., 11 de marzo de 2008.

II

Índice general

Introducción

Las votaciones tienen una gran importancia en la vida política de cualquier país democrático, también tienen relevancia en grupos en donde se eligen dirigentes a través de este medio. Dependiendo de la importancia de la votación, se hace por medio de boletas en papel o por medio de votos vía email. La seguridad, legitimidad y privacidad de cada voto en cualquier caso están basadas en la confianza que se tiene en las personas involucradas directamente en el proceso de recolección y escrutiño de los votos, con sus ventajas y desventajas en cada caso anterior (anonimato del voto otorgado por los votantes, no duplicidad de votos emitidos por cada votante, etc.).

El proceso de votación se puede transladar al medio electrónico asegurnado el anominato de los votantes y la emisión de votos (emisión a lo más de un único voto por votante) por medio de elementos criptográficos.

En el presente trabajo se aborda el tema de votación electrónica y se describe su seguridad criptográfica. En el capítulo 1 se da una introducción a los conceptos criptográficos necesarios para el entendimiento de los elementos involucrados en la votación electrónica. En el capítulo **??** se abarca diferentes tipos de elecciones: convencionales, electrónicas y electrónicas con uso de criptografía. En el capítulo 2 se describen diferentes protocolos criptográficos auxiliares que sirven para garantizar la seguridad criptográfica y anonimato de los votantes. En el capítulo 3 se describen dos diferentes protocolos de vatación electrónica. Finalmente se muestran las conclusiones del presente trabajo.

Capítulo 1

Conceptos Criptográficos

Este capítulo aborda los principios de lo que se conoce como criptografía, así como conceptos basicos tales como esquemas de firmado y cifrado, primitivas criptográficas, firmas ciegas, que son la herramienta básica para el desarrollo y entendimiento de la tesis, porque será la base para asegurar una votación electrónica segura y confiable.

1.1. Introducción a la Criptografía

Esta sección describe la historia de la Criptografía, a través de los tiempos como se fue perfeccionando este utensilio para una mejor seguridad al cifrarlo, se utilizaban los aparatos sofisticados solo aquellos que conocían sobre esto que llamaban Criptografía.

En el año 500 a.C. los griegos utilizaron un cilindro llamado *scytale* alrededor del cual enrollaban una tira de cuero. Al escribir un mensaje sobre el cuero y desenrollarlo se veía una lista de letras sin sentido. El mensaje correcto sólo pudo leerse al enrollar el cuero nuevamente en un cilindro de igual diámetro.

Durante el Imperio Romano Julio César empleo un sistema de cifrado el cual consistia en sustituir la letra a cifrar por otra letra distanciada a tres posiciones más adelante. Durante su reinado, los mensajes de Julio César nunca fueron decifrados.

En el S. XII Roger Bacon y en el S. XV León Batista Alberti inventaron y publicaron sendos algoritmos de cifrado basados en modificaciones del método de Julio César.

Durante la segunda guerra mundial en un lugar llamado Bletchley Park[1] un grupo de científicos trabajaba en Enigma, la máquina encargada de cifrar los mensajes secretos alemanes.

En este grupo se encontraban tres matemáticos polacos llamados Marian Rejewski, Jerzy Rozycki, Henryk Zygalski y un joven retraído llamado Alan Turing quien había sido reclutado debido a que unos años antes había creado una computadora binaria.

Probablemente poca gente en los servicios secretos ingleses sabía lo que era un computadora (y mucho menos binaria) pero no cabía duda que sólo alguien realmente inteligente podía inventar algo así, cualquier cosa que eso fuese.

Era mucho más abstracto que todos sus antecesores y sólo utilizaba 0 y 1 como valores posibles de las variables de su álgebra. Sería Turing el encargado de descifrar el primer mensaje de Enigma y cambiar el curso de la guerra, de la historia y de la Seguridad Informática actual.

1.2. Conceptos Básicos

Esta sección describe algunas conceptos como son el caso de mensaje llano, canal seguro, receptor, mensaje cifrado entre muchas otros que serviran,para el entendimiento y comprensión de conceptos que se utilisarán durante la tesis.

A continuación se da una noción de términos y conceptos básicos. Las correspondientes definiciones se detallarán en las secciones

Alfabeto : Conjunto finito de símbolos que se utilizan para plasmar información en mensajes. Al alfabeto en uso lo denotaremos por Σ. Con frecuencia utilizaremos un alfabeto binario, es decir, $\Sigma = \{0, 1\}$.

Mensaje o Documento : Secuencia de elementos del alfabeto.

Mensaje Original : Mensaje creado por el emisor.

Mensaje Llano : Mensaje al cual no se le ha aplicado ningún tipo de transformación.

[1] lugar situado a unos 70 Km al norte de Londres

Mensaje Cifrado : Resultado obtenido al aplicar una transformación de cifrado a un mensaje llano.

Mensaje Descifrado : Resultado obtenido al aplicar una transformación de descifrado a un mensaje cifrado.

Firma Digital : Resultado obtenido al aplicar una transformación de firmado a un mensaje llano.

Verificación de Firma : Resultado obtenido al aplicar una transformación de verificación a un mensaje firmado.

Función de Resumen : Función $H : \Sigma^m \to \Sigma^n$, donde $m > n$.

Emisor : Entidad que genera un mensaje y le aplica una transformación.

Receptor o Destinatario : Entidad que recibe un mensaje por parte del emisor.

Canal : Medio a través del cual se envía un mensaje.

Seguridad : Transformación que se le aplica a un mensaje para que la información contenida en éste quede confidencial y/o íntegra y/o auténtica y/o sea imputable a su emisor.

Canal Seguro :Donde la información permanece confidencial, es decir, que solo el emisor y receptor pueden conocer el contenido del mensaje original enviado a través de este canal.

Canal Inseguro :Donde la información puede ser vista, interceptada, modificada, etc., sin el conocimientos ni autorización del emisor y/o receptor.

Criptogragía : Área interdisciplinaria que persigue el dotar de seguridad a mensajes.

Protocolo Criptográfico : Es un protocolo (es decir un conjunto bien definido de etapas, implicando a dos o más partes y por ellas acordado, designado para realizar una tarea específica) que utiliza como herramienta algún algoritmo criptográfico.

1.3. Objetivo de la Criptografía

Esta sección describe lo que significa etimologicamente hablando la palabra Criptografía, asi tambien saber de donde deriva o es rama de alguna ciencia, al igual que se maneja la definición en si. También se menciona el por que utilizar en este caso particular la Criptografía, describiendo las caracteristicas, objetivos y otras cuestiones en las que se basa la Criptografía.

La palabra Criptografía proviene etimológicamente del griego $\kappa\rho\upsilon\pi\tau o\varsigma$ (kryptos) $\gamma\rho\alpha\phi\omega$ (grafo) y significa arte de escribir con clave secreta o de un modo enigmático. Aportando luz a la definición cabe aclarar que la Criptografía hace años que dejó de ser un arte para convertirse en una técnica (o conjunto de ellas) que tratan sobre la protección (ocultamiento ante personas no autorizadas) de la información.

Entre las disciplinas que engloba cabe destacar la Teoría de la Información, la Matemática Discreta, la Teoría de los Grandes Números y la Complejidad Algorítmica. Es decir que la Criptografía es la ciencia que consiste en transformar un mensaje inteligible en otro que no lo es (mediante claves que sólo el emisor y el destinatario conocen), para después devolverlo a su forma original, sin que nadie que vea el mensaje cifrado sea capaz de entenderlo.

La importancia de la Criptografía radica en que es el único método actual capaz de hacer cumplir el objetivo de la Seguridad Informática, mantener la privacidad, integridad, autenticidad y hacer cumplir con la *no repudiación de origen* o *imputabilidad*, relacionado a no poder negar la autoría y recepción de un mensaje enviado.

Los servicios que provee la criptografía en dicho escenario son confidencialidad, integridad de información, autenticación de origen de datos y de las entidades participantes y no repudio. Dichos servicios pueden ser definidos como sigue:

1. *Confidencialidad* La confidencialidad asegura que la información sensible sólo podrá ser consultada o manipulada por usuarios, entidades o procesos autorizados. Existen algoritmos matemáticos para proveer la confidencialidad haciendo ilegible la información.

2. *Integridad* La integridad da la certeza de que la información no ha sido modificada por entidades no autorizadas para hacerlo. Dentro de las posibles modificaciones están la escritura, modificación o borrado de segmentos de datos.

3. *Autenticación* La autenticación asegura que la identidad de los participantes es verdadera. La autenticación se logra verificando dichas identidades usan-

do mecanismos como firmas digitales, certificados digitales o características biométricas. El no contar con este servicio compromete a los servicios antes mencionados dado que cualquier entidad o usuario no autorizado puede realizar algún ataque a la seguridad.

En el caso de los usuarios se pueden evaluar tres aspectos para autenticarlos, el primer aspecto es verificar algo que el usuario tiene, por ejemplo una clave privada con la cual puede emitir firmas digitales; el segundo aspecto es poner a prueba al usuario sobre algo que sabe, esto es, pedirle una contraseña y, finalmente, el tercer aspecto es verificar algo que el usuario es, por ejemplo, analizar sus huellas dactilares o su retina.

4. *Imputabilidad* La no repudiación ofrece protección a un usuario o entidad frente a que otro usuario niegue posteriormente que en realidad se realizó cierta transacción. Esta protección se efectúa por medio de una colección de evidencias irrefutables que permitirán la resolución de cualquier disputa.

Los objetivos de la seguridad se enlistan:

Confidencialidad	Mantener la información en secreto y algunos pueden verla
Integridad de datos	Asegurar que la información no ha sido alterada
Autenticación	Corroborar la identidad de una entidad
Autenticación de mensajes	Corroborar la fuente de la información
Firmas	Mecanismo para ligar información con una entidad
Autorización	Permitir a una entidad realizar algo sobre información
Validación	Proporcionar periodos autorización para utilizar información
Control de Acceso	Privilegios de acceso a recursos restringidos para entidades
Certificación	Endoso de información por una entidad en la cual se confía
Estampas de tiempo	Registro de la fecha de creación o la existencia de ella
Atestiguar	Verificar la creación de la información por una entidad externa
Recibo	Reconocimiento de que se ha recibido la información

Confirmación	Reconocimiento de que los servicios se han proporcionado
Propiedad	Dar a la entidad el derecho legal de utilizar un recurso con otras
Anonimato	Encubrir la identidad de una entidad implicada en algunos procesos
No repudio	Prevención de la negación de una entidad sobre acciones
Revocación	Anular la certificación o la autorización a una entidad

1.4. Esquemas de Cifrado

Esta sección describe detalladamente los principales esquemas de cifrado que se tienen, que son el simétrico, donde se maneja una clave secreta conocida unicamente por las dos partes y el asimétrico (o de clave pública) que utiliza dos claves distintas pero independientes. Los esquemas de cifrado se manejan para lograr una forma de asegurar la confiabilidad en la elección.

Entre las necesidades que se intentan satisfacer actualmente, mediante el cifrado de mensajes a fin de garantizar la confidencialidad, pero también la autenticación en un entorno que elimina toda presencia física, la firma de documentos digitales inmateriales, la garantía de integridad de las transmisiones a través de las redes abiertas amenazadas por múltiples piratas. En general, se distinguen dos grandes esquemas en materia de cifrado: el cifrado convencional y el cifrado de clave pública.

El primero, denominado cifrado convencional o cifrado simétrico, se basa en la hipótesis de que una clave secreta, una simple serie de bits, es inicialmente compartida por los interlocutores. Un esquema de cifrado se define como un algoritmo parametrizado por la clave que tiene la propiedad de transformar un mensaje llano en un mensaje cifrado de manera tal que la transformación inversa sea también fácil si se conoce la clave pero difícil si no se le conoce.

El algoritmo de cifrado simétrico más conocido es sin duda la norma comercial americana DES adoptada en 1976. Pone en práctica los principios clásicos de difusión y de confusión de datos formalizados por Shannon desde 1949, y de manera sorprendentemente sólida, como lo han demostrado los veinte años de tentativas infructuosas de ataque. DES ha sido un incentivo formidable para los criptoanalistas y para los diseñadores de criptosistemas.

Actualmente se usan algoritmos que utilizan claves de un tamaño que va de 112 a 256 bits, tal como el AES (Advanced Encryption Standard). La seguridad frente a los ataques exhaustivos es entonces colosal: un número como 2 elevado a la potencia 256 es comparable con el número de electrones estimados del universo.

Sin embargo, un problema inherente a la criptografía simétrica persiste aún: ¿qué procedimiento seguir para ponerse de acuerdo inicialmente sobre una clave secreta compartida? Esto no causa mayores problemas en el caso de aplicaciones limitadas a pequeñas redes que conectan pocos participantes, pero es fundamental en el caso de grandes redes abiertas como Internet.

Los sistemas de cifrado asimétricos, llamados también de clave pública, hacen uso de dos claves diferentes. Una es la clave pública, la cual puede darse a conocer a cualquier persona; y la otra, llamada clave privada, la cual se debe mantener en secreto. Para enviar un mensaje, el que envia el mensaje usa la clave pública del que recibira el mensaje para cifrar el mensaje.

Una vez que el mensaje ha sido cifrado, únicamente con la clave privada del que recibio el mensaje se podrá descifrar, ni siquiera la entidad que originalmente cifró el mensaje puede volver a descifrarlo. Por ello, se puede dar a conocer sin problema la clave pública para que todo aquel que se quiera comunicar confidencialmente con el destinatario lo pueda hacer. Este tipo de sistemas ofrecen de manera óptima los servicios de seguridad de datos, autenticación de entidades y de datos, y el no repudio.

1.5. Esquemas de Firma Digital

Esta sección describe como crear una firma digital, la cual se usa como prueba de fidelidad del mensaje a manejar. En la sección se especifica las caracteristicas a cumplir de la firma, el procedimiento para realizar una firma digital, al igual que se describirá, como verificarla para demostrar autenticidad de la firma digital

La firma digital es un conjunto o bloque de caracteres que viaja junto a un documento, archivo o mensaje y que puede acreditar quién es el autor o emisor del mismo (lo que se denomina autenticación), y que nadie ha manipulado o modificado el mensaje en el transcurso de la comunicación. La firma digital equivale funcionalmente a la firma autógrafa en cuanto a la identificación del autor del que procede el mensaje. La aparición y desarrollo de las redes de computadoras, de las que Internet es el ejemplo más notorio, ha supuesto la posibilidad de intercambiar entre personas

geográficamente distantes todo tipo de mensajes, incluidos los de contenido privado.

Estos mensajes plantean el problema de acreditar tanto la autenticidad como la autoría de los mismos. Concretamente, para que dos individuos puedan intercambiarse entre sí mensajes electrónicos de carácter privado que sean mínimamente fiables y puedan, en consecuencia, dar a las partes involucradas la confianza y la seguridad que necesita el tráfico de dicho tipo de mensajes, se deben cumplir los siguientes requisitos:

1. Autenticación, que implica poder atribuir de forma indudable el mensaje electrónico recibido a una determinada persona como autora del mensaje.

2. Integridad, que implica la certeza de que el mensaje recibido por el receptor es exactamente el mismo mensaje emitido por el emisor, sin que haya sufrido alteración alguna durante el proceso de transmisión de emisor hacia receptor.

3. Imputabilidad, no repudiación de origen, que implica que el emisor del mensaje no pueda negar en ningún caso que el mensaje ha sido enviado por él.

El procedimiento técnico de una firma digital, basándose en técnicas criptográficas trata de dar respuesta a esa triple necesidad apuntada anteriormente, a fin de posibilitar el tráfico comercial electrónico.Para darse lo anterior la firma digital, en algunos casos puede tener mucho más pasos o en un acomodo diferente al que a continuacion se sugiere como estructura general para poder firmar.

Generación de Claves en este paso se genera el par de claves para la firma digital, es decir:

$$Gen(1^S) \rightarrow (SK, PK)$$

Donde Gen se usa para decir que se esta generando la firma, 1^s es el espacio de $0, 1$, SK es la clave privada o secreta de firmado y PK es la clave pública o de verificación.

Creación de Firma donde se va a originar nuestra firma mediante:

$$Sig(M, SK) \rightarrow \sigma$$

Donde Sig se usa para describir la firma del mensaje y clave secreta, M es el mensaje y σ es la firma digital

Verificación de Firma Teniendo la firma para verificar la autenticidad de esta se debe:

$$Ver(M, \sigma, PK) \rightarrow F \text{ ó } V$$

Donde Ver se usa para decir que se esta verificando la firma para dar a conocer si es verdadera o falsa.

Siendo un esquema generalizado y solo mencionado ahora se vera un ejemplo más en particular en el cual se deberá utilizar dos técnicas distintas, que son la criptografía asimétrica o de clave pública para cifrar los mensajes y el uso de las llamadas funciones hash o funciones resumen.

En ocasiones, además de garantizar la procedencia de los mensajes electrónicos que se intercambian por medio de internet y la autenticidad o integridad de los mismos, puede ser conveniente garantizar también su confidencialidad, que no es un requisito esencial de la firma digital sino accesorio de la misma. Ello implica que el mensaje no pueda ser leído por terceras personas distintas del emisor y del receptor durante su proceso de transmisión, es decir, que se debe tener la certeza de que el mensaje enviado por el emisor únicamente será leído por el receptor y no por personas ajenas a la relación que mantienen emisor y receptor.

En tales casos, también se acude al cifrado del mensaje con el par de claves, pero de manera diferente al mecanismo propio y característico de la firma digital. Para garantizar la confidencialidad del mensaje, el cuerpo del mismo (no el hash o resumen) se cifra utilizando la clave pública del receptor, quien al recibir el mensaje lo descifrará utilizando para ello su clave privada (la clave privada del receptor). De esta manera se garantiza que únicamente el receptor pueda descifrar el cuerpo del mensaje y conocer su contenido.

1.6. Funciones de Resumen

Esta sección describe una herramienta que son las funciones hash, que tienen la caracteristica especial de estar definidas de espacios de longitud arbitraria a espacios de longitud fija. Se describe a las funciones hash y las caracteristicas que deben tener.

Los mensajes que se intercambian pueden tener un gran tamaño, hecho que dificulta el proceso de cifrado. Por ello, en la practica no se cifra el mensaje entero sino un resumen del mismo, el cual se obtiene aplicando al mensaje una función hash.

Partiendo de un mensaje determinado que puede tener cualquier tamaño, éste se convierte mediante la función hash en un mensaje con una dimensión fija.

Para ello, el documento original se divide en varias partes, cada una de las cuales tendrá el mismo tamaño, y una vez dividido se combinan elementos tomados de cada una de las partes resultantes de la división para formar el resumen, que también tendrá una dimensión fija y constante. Este resumen del mensaje es el que se cifrará utilizando la clave privada del emisor del documento.

Las propiedades matemáticas que se quieren para una función hash son las siguientes:

- La función h debe ser fácil de calcular para cualquier m.
- La función hash debe ser **resistente a pre-imagenes** (de un sólo sentido), es decir, si se conoce $h(m)$ encontrar una pre-imagen es computacionalmente imposible.
- La función hash debe ser **resistente a las colisiones**, es decir, no debe ser posible (computacionalmente) encontrar m y m' con $m \neq m$ tales que $h(m) = h(m')$.

Las funciones hash son usadas principalmente para resolver problemas relacionados con la integridad de los mensajes, así como en los procesos de verificación de la autenticidad de mensajes y de su origen.

También son utilizadas en criptosistemas de clave pública para cifrar la clave privada mediante una contraseña elegida por el dueño del par de claves. Esto es, se le pide al usuario que indique una contraseña (alrededor de 10 caracteres), y después se le aplica una función hash; el resumen resultante funciona como clave simétrica para cifrar la clave privada a través de un algoritmo simétrico ($DES, TripleDES, etc$). Y así, el usuario puede cifrar y descifrar su clave privada con sólo recordar la contraseña inicial que proporcionó.

Entre los algoritmos más importantes de las funciones hash están: $MD5$ y $SHA-1$. El Algoritmo $MD5$ es el resultado de una serie de mejoras sobre el algoritmo $MD4$, diseñado por Ron Rivest, el cual procesa los mensajes de entrada en bloques de $512bits$, y produce una salida de $128bits$. El algoritmo $SHA-1$, roto recientemente, fue desarrollado por la NSA, para ser incluido en el estándar DSS (Digital Signature Standard). Produce cadenas de 160 bits, a partir de bloques de 512 bits del mensaje original.

1.7. Anonimato y Pseudoanonimato

Esta sección describe lo que es especificamente el anonimato y donde se presenta con mayor frecuencia, así como se mencionan ejemplos cotidianos. Se debe recordar que el anonimato es indispensable ya que el voto debe ser anonimo por eso es muy importante tener confidencialidad para que se de una votación electrónica adecuada.

Así como también se describirá con sumo detalle dos formas de anonimato como son la firma ciega y las monedas electrónicas que en su momento se verá la importancia de estas, al igual que se mencionan sus características.

El anonimato es el estado de una persona siendo anónima, es decir, que la identidad de dicha persona es desconocida. Esto puede ser simplemente porque no se le haya pedido su identidad, como en un encuentro ocasional entre extraños, o porque la persona no puede o no quiere revelar su identidad. Por ejemplo, esto puede suceder a víctimas de crimen y guerra, cuya identidad no puede ser reconocida. Disfrazar la identidad de uno puede también ser por elección, por razones legítimas como la privacidad y, en algunos casos, seguridad personal. Los criminales a menudo prefieren mantener su anonimato, como en el caso de escribir una carta con una amenaza o demanda.

En una gran ciudad existe mayor anonimato que en pequeños poblados. Esto puede ser considerado una ventaja o desventaja. Los trabajos anónimos no poseen un autor conocido. Pueden ser el resultado de una tradición folklórica, la difusión oral; o puede tratarse de que la información del autor fue extraviada o escondida intencionalmente. Hablando del anonimato cibernético, cuando las personas toman parte en un chat room, o contribuyen con algo, participan, etc., ejercen un control directo sobre los elementos de su autorepresentación. Al elegir nombres, descripciones personales, etc., deciden conscientemente cómo quieren que los demás los perciban. Todo esto podría verse, como afirman A. F. Wood/M. J. Smith como un continuo de identificación que iría desde la identidad en la vida real al anonimato pasando por el pseudoanonimato.

El servicio de anonimato actúa como un filtro de seguridad entre tu navegador y el sitio Web que deseas visitar. Te conectas al anonimizador, introduces el URL al que deseas ir, entonces éste se adentra en la Red en busca de la página que deseas ver y te la muestra. Si posteriormente vas siguiendo enlaces de una página a otra, se presentarán así mismo a través del anonimizador. Sus inconvenientes:

- No funcionan con todos los sitios ni con los servidores seguros.

- Tampoco se reciben cookies (lo cual para algunos representa más bien un alivio).
- Desactivan todos los programas en Java, JavaScript, etc. (de nuevo, ventaja o inconveniente según para quién).
- Hace más lenta la navegación.
- Para un servicio óptimo hay que pagar.
- Añaden a las páginas que visitamos banners con publicidad de sus patrocinadores.

El servicio de navegación anónima más conocido es Anonymous Surfing.

Normalmente se asocia el contenido "secreto" al contenido secreto de noticias ya que en muchas situaciones es deseado, que el intercambio de noticias entre participantes permanezca secreto en este caso se habla de **anonimato** se puede diferenciar tres clases de anonimato:

- Anonimato del emisor
- Anonimato del receptor
- Anonimato en la relación de comunicación

En el último caso deben permanecer el receptor y el emisor en secreto uno con el otro y con terceros; el anonimato del receptor puede ser relativamente fácil alcanzarlo a través del "broadcasting" para ello las noticias pueden ser mandadas, a pesar de que solo es hecho para uno. Nos ocuparemos primordialmente del anonimato del emisor.

1.7.1. Firma ciega

En esta sección se explicara lo que es una firma ciega y como se puede elaborar una de ellas al igual que un ejemplo muy claro de lo que se puede entender como firma ciega.

Normalmente el firmante querrá saber el contenido de lo que firma. Esto también se tiene en el caso de una firma digital, sin embargo hay situaciones en la que no es requerido, de hecho es exigido, que el firmante no sepa que es lo que firma. Ejemplos de esta situación son monedas electrónicas y elecciones electrónicas.

Un procedimiento para generar firmas ciegas electrónicas es un protocolo, en el cual una persona A obtiene una firma en un documento hecha por una persona B, que cumple:

- B no puede ver el mensaje y
- A obtiene una firma auténtica de B.

Se describirá el mecanismo de una firma ciega en un ejemplo cotidiano y después en un protocolo criptográfico.

María obtuvo una mala calificación en su trabajo escolar, el cual debe ser firmado por su padre. Ella cree que no volverá a pasar y no quiere que su papá se entere ni que la regañe, ya que María siempre tiene buenas calificaciones. Le hace a su papá el siguiente juego:

- María mete su trabajo en un sobre junto con un papel calca y le enseña a su padre el lugar donde debe firmar (en el sobre).
- El padre firma, su firma se calca y así el trabajo escolar queda firmado.

Para asegurarse que no se ha firmado ningún cheque, se puede hacer una pequeña anotación; por ejemplo con una anotación diciendo "reporte visto".

Con ayuda del procedimiento de RSA se puede realizar esta idea criptográficamente, esto se ve en el procedimiento de [Cha85] B debe hacer una firma ciega a un documento m de una persona A. Sea n el módulo, e la clave pública y d la clave privada del firmante B.

A elige aleatoriamente un número r que es divisor de n saca la potencia con e y manda el valor

$$x = mr^e \mod n$$

a B (la multiplicación con r^e lo utilizamos de la misma manera en que María utilizó el sobre). B firma el valor x y envía el resultado

$$y = x^d \mod n$$

a A. A multiplica el valor resultante con la inversa de r y obtiene

$$yr^{-1} = x^d r^{-1} = (mr^e)^d r^{-1} = m^d r^{ed} r^{-1} = m_d r r^{-1} = m^d \mod n$$

que es el mensaje firmado.

La firma así obtenida es ciega, puesto que B no sabe cual es en realidad el documento que firmó, debido a que el número m que se multiplica por r^e no puede obtenerse directamente de ellos.

1.7.2. Monedas electrónicas

En esta sección se describe la manera de usar las monedas comunes y corrientes, comparandolas con monedas electrónicas y estas a su vez una breve descripción de como se usarian en el proceso.

La forma más importante en que nos enfrentamos al anonimato es el pago con monedas. Este paso nos es común, pero por lo general no sabemos que se trata de un proceso de anonimato. Contrario al pago con cheques, no se puede reconocer quién nos ha dado una moneda.

Se formula entonces la pregunta: ¿es posible hacer lo mismo con el pago electrónico? Sí, de hecho se puede utilizar "monedas electrónicas" con la ayuda de *firmas ciegas* que garantizan mayor grado de anonimato.

Aún en el caso más sencillo se divide el ciclo de una moneda en al menos tres instancias: **banco central**, el cual hace y distribuye las monedas, **el cliente** que recibe las monedas del banco y éste lo usa para comprar bienes y servicios, **el comerciante** que toma las monedas y las cambia en el banco. Por sencillez supongamos que hay un solo banco que expide monedas y administra las cuentas de clientes y comerciantes.

Un sistema de pagos debe de satisfacer las necesidades de seguridad de todos los participantes, las cuales son entre otras:

- Generación central: solo el banco tiene permitido la emisión de monedas
- Autenticidad: todos los participantes deben poder probar la autenticidad de las monedas
- Anonimato: cuando el banco expide monedas no debe reconocer a quién le fueron dadas. El comerciante debe aceptar una moneda sin que el cliente tenga que identificarse.
- Claridad: no se debe poder duplicar monedas, como tampoco debe ser posible utilizar dos veces una moneda.
- Para cada valor de una moneda el banco tiene una clave secreta, en este ejemplo particular se utilizará el esquema de firma de RSA.
- Uno puede reconocer las monedas auténticas verificando la clave pública que se envía con el valor de la moneda, cuyo valor tiene una estructura predeterminada (esquema de redundancia).

Generación de monedas: en el modelo de D. Chaum [Cha85] el banco no puede crear monedas anónimas solo, sino tiene que estar con un cliente. Imaginemos que el cliente quiere obtener una moneda de un Euro para ello elige un valor w (por ejemplo w puede ser cortado en dos mitades idénticas v: el cliente elige $v = 1234567$ y forma $w = 12345671234567$) el cliente obtiene una moneda m a la vez como w en el banco con la clave secreta asociada a un Euro.

Uso de una moneda: el cliente lleva la moneda con el comerciante, este último prueba su autenticidad usando la clave pública de un Euro. El acepta la moneda cuando el resultado de la prueba de redundancia es positiva.

Intercambio de monedas: el comerciante lleva las monedas al banco y este prueba la autenticidad de la moneda tal y como lo hizo el comerciante. El anonimato se obtiene del hecho de utilizar una firma ciega.

Este sistema de pago electrónico tiene el problema, entre otros, que la claridad no está resuelta de fondo, hay una gran variedad de propuestas de solución de naturaleza criptográfica y no criptográfica que hacen el doble de riesgoso el usar una moneda.

Bajo las posibles soluciones no criptográficas se tiene la propuesta de que las monedas electrónicas inmediatamente y en línea se deben intercambiar en el banco, lo que naturalmente para el comerciante se traduce en altos costos.

Nuevas investigaciones en este tema se ocupan de la pregunta ¿pueden las monedas electrónicas intercambiarse igual que las genuinas en lugar de que se tengan que intercambiar inmediatamente en el banco? El gran problema es encontrar al culpable en la cadena por donde pasó una moneda, en el caso de una doble emisión de la misma.

1.8. Nociones de seguridad

Esta sección describe la manera de asegurarse el periodo de tiempo de resolución de nuestros sistemas, lo que se muestra es una explicación de las formas de medir la complejidad de los problemas y como se definen estas cuestiones.

En análisis de algoritmos una cota superior asintótica es una función que sirve de cota superior de otra función cuando el argumento tiende a infinito. Usualmente se utiliza la notación de Landau $O(g(x))$ para referirse a las funciones acotadas superiormente por la función $g(x)$.

Definición 1.1. *Dadas dos funciones* $f, g : \mathbb{R} \to \mathbb{N}$ *se dice* $f(x) = O(g(x))$ *si existen* $c, x_0 > 0$ *tales que* $\forall x \geq x_0$ *se cumple* $0 \leq |f(x)| \leq c|g(x)|$

Se consideran funciones con rango R^+ pues el tiempo de ejecución no puede ser negativo. Sólo interesa lo que pase asintóticamente ($n \geq n_0$), por lo que podríamos considerar funciones no definidas en todo N o que tomen valores negativos para algunos valores de n.

Se dice "$f(n)$ es $O(g(n))$", si $f(n)$ es como máximo una constante de tiempo $g(n)$, excepto posiblemente para algunos valores pequeños de n.

Se dice $f(n)$ es $O(g(n))$ si existe un entero n_0 y una constante $c > 0$ tal que para todos los enteros $n \geq n_0$ se tendrá que $f(n) \leq c * g(n)$ (cota superior).

La notación $f(n) = O(g(n))$ significa que $|f(n)| \leq c|g(n)|$ para $n \geq n_0$.

Por consiguiente, $|g(n)|$ es un límite o cota superior para $|f(n)|$.

Formalmente: $f(n) = O(g(n))$ si existen constantes $c \geq 0$ y n_0 tales que para $n \geq n_0$, es decir si:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c$$

Propiedades de la notación O

1. $c * O(f(n)) = O(f(n))$
2. $O(f(n)) + O(f(n)) = O(f(n))$
3. $O(f(n)) + O(g(n)) = O(f(n) + g(n))$
4. $O(O(f(n))) = O(f(n))$
5. $\text{Max}(O(f(n)), O(g(n))) = O(\text{Max}(f(n), g(n)))$
6. $O(log_a(n)) = O(log_b(n))\text{para}a, b > 1$
7. $O(log(n!)) = O(n * log(n))$
8. Para $k > 1$
$$O\left(\sum_{i=1}^{n} i^k\right) = O\left(n^{k+1}\right)$$
9. $$O\left(\sum_{i=1}^{n} i^{-1}\right) = O\left(\sum_{i=1}^{n} \frac{1}{i}\right) = O\left(\log(n)\right)$$

10. En general para todo $c > 0$ y a diferente de b :

$$O\left(c^{\log_a(n)}\right) \neq O\left(c^{\log_b(n)}\right)$$

11.

$$O\left(\sum_{i=2}^{n} \log(i)\right) = O\left(n \log(n)\right)$$

Ejemplo

La función $x^2 + 200x$ está acotada superiormente por x^2. Quiere decir que cuando x tiende a infinito el valor de $200x$ se puede despreciar con respecto al de x^2.

Ejemplo

Buscar en un diccionario tiene complejidad logarítmica. Se puede iniciar la búsqueda de una palabra por la mitad del diccionario. Inmediatamente se sabe si se ha encontrado la palabra o, en el caso contrario, en cuál de las dos mitades hay que repetir el proceso (es un proceso recursivo) hasta llegar al resultado. En cada (sub)búsqueda el problema (las páginas en las que la palabra puede estar) se ha reducido a la mitad, lo que se corresponde con la función logarítmica. Este procedimiento de búsqueda (conocido como búsqueda binaria) en una estructura ordenada tiene complejidad logarítmica $O(lnn)$.

Ejemplo

El proceso más común para ordenar un conjunto de elementos tiene complejidad cuadrática. El procedimiento consiste en crear una colección no vacía de elementos. A ella se añade, en orden, el menor elemento del conjunto original que aún no haya sido elegido, lo que implica hacer un recorrido completo del conjunto original ($O(n)$, siendo n el número de elementos del conjunto). Este recorrido sobre el conjunto original se realiza hasta que todos sus elementos están en la secuencia de resultado. Se puede ver que hay que hacer n selecciones (se ordena todo el conjunto) cada una con un coste de ejecución n : el procedimiento es de orden cuadrático $O(n^2)$. Hay que aclarar que hay diversos algoritmos de ordenación con mejores resultados.

1.8.1. Tiempo polinomial

Definición 1.2. *Se llama **algoritmo** a todo proceso bien definido que a partir de una serie de valores de entrada proporciona un conjunto de valores de salida.*

Definición 1.3. *Diremos que un* ***algoritmo*** *es* ***determinístico*** *si el valor de entrada determina por completo al de salida.*

Definición 1.4. *Si, por el contrario, al ejecutar el algoritmo con el mismo valor de entrada pueden obtenerse distintos valores de salida, el* ***algoritmo*** *se dice* ***probabilístico*** *o* ***aleatorio****.*

Definición 1.5. *Diremos que una* ***función*** f *es operable si existe un algoritmo que tomando un argumento x como entrada devuelve el valor $f(x)$ como salida.*

Nota:

Si A es un algoritmo, $A(x)$ denota al valor o valores de salida que proporciona A tras ejecutarse con valor de entrada x.

Los valores de entrada y salida de un algoritmo están codificados como secuencias de ceros y unos (bits). Supondremos sin pérdida de generalidad que cuando dichos valores son números naturales, su codificación se corresponde con su expresión en base dos.

Dado un valor cualquiera x, que un cierto algoritmo toma como entrada, llamaremos longitud de x a la longitud de su codificación en binario. La eficiencia de un algoritmo se mide por el llamado tiempo de ejecución, que va a medirse en función de la longitud de su entrada.

Definición 1.6. *El* ***tiempo de ejecución de un algoritmo*** *es el máximo número de operaciones elementales (operaciones bit a bit) que realiza al ejecutarse a partir de un cierto valor de entrada.*

El tiempo de ejecución de un algoritmo se expresará como una función de la longitud del valor de entrada (es decir, será una función definida sobre IN). Dichas funciones suelen expresarse a través de formulaciones asintóticas, que permiten clasificar los algoritmos correspondientes en base a su complejidad.

La complejidad en tiempo de un problema es el número de pasos que toma resolver una instancia de un problema, a partir del tamaño de la entrada utilizando el algoritmo más eficiente a disposición. Intuitivamente, si se toma una instancia con entrada de longitud n que puede resolverse en n^2 pasos, se dice que ese problema tiene una complejidad en tiempo de n^2. Por supuesto, el número exacto de pasos depende de la máquina en la que se implementa, del lenguaje utilizado y de otros factores. Para no tener que hablar del costo exacto de un cálculo se utiliza la notación O. Cuando un problema tiene costo en tiempo $O(n^2)$ en una configuración de

computador y lenguaje dado este costo será el mismo en todos los computadores, de manera que esta notación generaliza la noción de coste independientemente del equipo utilizado.

Uno de los problemas presentes en los conjuntos y los elementos es el ordenamiento para esto se se tienen estre otros dos métodos para poder organizar los elementos de manera diferente, esto es en el sistema quicksort lo que hace es acomodar u organizar los elementos que estan desordenados y dispersos en el conjunto para ordenarlos de primera instancia y la más común de menor a mayor, pero tambien se puede realizar en sentido contrario, es decir, de mayor a menor. Mientras que el otro sistema el bubblesort realiza tambien un acomodo de los elementos, pero este a comparación del quick el bubble compara los elementos tomando uno de ellos y acomandolos de manera que tenga un sentido de jerarquia en los elementos, al igual que el sistema de quick los ordena de menor a mayor o viceversa.

Ejemplo

Quicksort

El ordenamiento rápido (quicksort en inglés) es un algoritmo basado en la técnica de divide y vencerás, que permite, en promedio, ordenar n elementos en un tiempo proporcional a n log n. Esta es la técnica de ordenamiento más rápida conocida. Fue desarrollada por C. Antony R. Hoare en 1960. El algoritmo original es recursivo, pero se utilizan versiones iterativas para mejorar su rendimiento (los algoritmos recursivos son en general más lentos que los iterativos, y consumen más recursos).

Descripción del algoritmo.

El algoritmo fundamental es el siguiente:

- Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote.
- Volver a situar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.
- La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.
- Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.

  Como se puede suponer, la eficiencia del algoritmo depende de la posición en la que termine el pivote elegido.

- En el mejor caso, el pivote termina en el centro de la lista, dividiéndola en dos sublistas de igual tamaño. En este caso, el orden de complejidad del algoritmo es $O(nlogn)$.
- En el peor caso, el pivote termina en un extremo de la lista. El orden de complejidad del algoritmo es entonces de $0(n^2)$. El peor caso dependerá de la implementación del algoritmo, aunque habitualmente ocurre en listas que se encuentran ordenadas, o casi ordenadas.
- En el caso promedio, el orden es $O(nlogn)$.

  No es extraño, pues, que la mayoría de optimizaciones que se aplican al algoritmo se centren en la elección del pivote.

Ejemplo

Bubblesort

El Ordenamiento de Burbuja (Bubble Sort en inglés) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas "burbujas". También es conocido como el método del intercambio directo.

Dado que solo usa comparaciones para operar elementos, se lo considera un algoritmo de comparación, siendo el más sencillo de implementar.

Una manera simple de expresar el ordenamiento de burbuja en pseudocódigo es la siguiente:

$$\text{procedimiento burbuja } (a_0, a_1, a_2, \ldots, a_{n-1})$$

$$\text{Para } i = 0 \text{hasta } n-2 \text{haga lo siguiente :}$$

$$\text{Para } j = i+1 \text{hasta } n-1 \text{ haga lo siguiente:}$$

$$\text{Si } a_i > a_j \text{entonces:}$$

$$(a_i, a_j) \leftarrow (a_j, a_i)$$

El ordenamiento de burbuja tiene una complejidad $\Omega(n)$. Cuando una lista ya está ordenada, el ordenamiento de burbuja pasará por la lista una vez, y encontrará que no hay necesidad de intercambiar las posiciones de los elementos. Por

ende, el ordenamiento hará solamente n comparaciones y determinará que la lista está completamente ordenada. Además, usará considerablemente menos tiempo que $O(n^2)$ si los elementos en la lista desordenada no están demasiado lejos de su ubicación correcta.

Se observa que los métodos utilizados para el ordenamiento aunque sirven con el mismo fin, estos no tienen el mismo procedimiento, ni tampoco los mismo acomodos de elementos por lo cual, dependiendo en lo que se use van a tener cada uno un rendimiento optimo. Sabiendo que el quick se debe utilizar para elementos en desorden muy marcado o disperso se tienen que usar este método, con lo que respecta al bubble este debe ser utilizado para obtener un resultado optimo en elementos que solo necesiten acomodarse con jerarquias, teniendo en cuanta las diferencias es como se tendran los resultados en cuanto a acomodos.

Definición 1.7. *Un **algoritmo se dice polinomial** si su tiempo de ejecución es $O(n^k)$, con k un número natural cualquiera. Todo algoritmo que no cumpla esta condición se dirá **exponencial**.*

Definición 1.8. *Diremos que una **función es computable en tiempo polinomial** si existe un algoritmo polinomial para su evaluación, en otro caso diremos que es computable en tiempo exponencial.*

En computación, cuando el tiempo de ejecución de un algoritmo (mediante el cual se obtiene una solución al problema) es menor que un cierto valor calculado a partir del número de variables implicadas (generalmente variables de entrada) usando una fórmula polinomial, se dice que dicho problema se puede resolver en un tiempo polinomial.

Por ejemplo, si determinar el camino óptimo que debe recorrer un cartero que pasa por N casas necesita menos de $50 * N^2 + N$ segundos, entonces el problema es resoluble en un "tiempo polinomial".

De esa manera, tiempos de $2 * n^2 + 5n$, o $4 * n^6 + 7 * n^4 - 2 * n^2$ son polonomial; pero 2^n no lo es.

Dentro de los tiempos polinomial, podemos distinguir los logarítmicos ($log(n)$), los lineales (n), los cuadráticos (n^2), cúbicos (n^3), etc.

Definición 1.9. *Llamamos **problema de decisión** a toda pregunta que depende de un cierto valor de partida y cuya respuesta es un valor en el conjunto $\{0, 1\}$ (falso o verdadero). Diremos que un problema de decisión D pertenece a la clase...*

- *...P (o que es polinomial) si existe un algoritmo que resuelve el problema D en tiempo polinomial.*

- *. . . NP si se sabe que el problema tiene solución y existe un algoritmo polinomial para verificar una solución de una instancia del problema.*

Un problema dado puede verse como un conjunto de preguntas relacionadas, donde cada pregunta se representa por una cadena de caracteres de tamaño finito. Por ejemplo, el problema factorización entera se describe como: Dado un entero escrito en notación binaria, retornar todos los factores primos de ese número. Una pregunta sobre un entero específico se llama una instancia. por ejemplo, "Encontrar los factores primos del número 15" es una instancia del problema factorización entera.
Los problemas de decisión se clasifican en conjuntos de complejidad comparable llamados clases de complejidad.

La mayor parte de los problemas en teoría de la complejidad tienen que ver con los problemas de decisión, que corresponden a poder dar una respuesta positiva o negativa a un problema dado. Por ejemplo, el problema "es primo" se puede describir como: Dado un entero, responder si ese número es primo o no. Un problema de decisión es equivalente a un lenguaje formal, que es un conjunto de palabras de longitud finita en un lenguaje dado. Para un problema de decisión dado, el lenguaje equivalente es el conjunto de entradas para el cual la respuesta es positiva.

1.8.2. P

Es el conjunto de los problemas de decisión que pueden ser resueltos en una máquina determinista en tiempo polinomial, lo que corresponde intuitivamente a problemas que pueden ser resueltos aún en el peor de sus casos.

Por ejemplo:

- Si un número T está escrito como un número binario, el algoritmo secuencial más obvio puede tomar tiempo 2^n. En cambio, si T está escrito como un número unario (una cadena de n unos, para $n = T$), solo toma tiempo n. Al escribir T en unario en lugar de binario, se reduce el algoritmo obviamente secuencial de tiempo exponencial a lineal.

- Existen tres minas de carbón cuya producción diaria es: La mina a produce 40 toneladas de carbón por día; La mina b otras 40 t/día; y, La mina c produce 20 t/día.

  En la zona hay dos centrales termoeléctricas que consumen: La central d consume 40 t/día de carbón; y, La central e consume 60 t/día

Los costos, de mercado, de transporte por tonelada son: De a a $d = 2$ monedas De a a $e = 11$ monedas De b a $d = 12$ monedas De b a $e = 24$ monedas De c a $d = 13$ monedas De c a $e = 18$ monedas

La solución de costo mínimo de transporte diario resulta: $X_{b-d} = 40$ resultando un costo de $12x40 = 480$ monedas $X_{a-e} = 40$ resultando un costo de $11x40 = 440$ monedas $X_{c-e} = 20$ resultando un costo de $18x20 = 360$ monedas

Total $1,280$ monedas.

- Para el cálculo del máximo común divisor de dos o más números se descompondrán los números en factores primos y se tomarán los factores comunes con su menor exponente. Por ejemplo, de las factorizaciones de 6936 y 1200,

$$6936 = 2^3 * 3 * 17^2$$

$$1200 = 2^4 * 3 * 5^2$$

podemos inferir que su m.c.d. es $2^3 * 3 = 24$ ó comunmente expresado como $mcd(6936, 1200) = 2^3 * 3 = 24$.

Si el número es muy grande este método no es operativo porque no conocemos los posibles factores. En ese caso tenemos que utilizar el algoritmo de Euclides.

- Definimos una máquina de Turing sobre el alfabeto 0,1, donde 0 representa el símbolo blanco. La máquina comenzará su proceso situada sobre un símbolo "1" de una serie. La máquina de Turing copiará el número de símbolos "1" que encuentre hasta el primer blanco detrás de dicho símbolo blanco. Es decir, situada sobre el 1 situado en el extremo izquierdo, doblará el número de símbolos 1, con un 0 en medio. Así, si tenemos la entrada "111" devolverá "1110111", con "1111" devolverá "111101111", y sucesivamente.

1.8.3. NP

En teoría de la complejidad computacional, *NP* es el acrónimo en inglés de Polinómial no determinista (Non-Deterministic Polynomial-time). Es el conjunto de problemas que pueden ser resueltos en tiempo polinómial por una máquina de Turing no determinista.

La importancia de esta clase de problemas de decisión es que contiene muchos problemas de búsqueda y de optimización para los que se desea saber si existe una cierta solución o si existe una mejor solución que las conocidas.

En esta clase están el problema del viajante (también llamado "problema del agente de ventas" o "problema del agente viajero") donde se quiere saber si existe una ruta óptima que pasa por todos los nodos en un cierto grafo y el problema de satisfacibilidad booleana en donde se desea saber si una cierta fórmula de lógica proposicional puede ser cierta para algún conjunto de valores booleanos para las variables.

El problema del agente viajero es un ejemplo. La respuesta al problema es conocida, es decir se conoce la forma de resolverlo, pero sólo en teoría, en la práctica la solución no es aplicable debido al tiempo que computacionalmente se precisa para obtener su resultado.

Sean N ciudades de un territorio. El objetivo es encontrar una ruta que, comenzando y terminando en una ciudad concreta, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida por el viajante. Es decir, encontrar una permutación $P = \{c_0, c_2, \ldots, c_{n-1}\}$ tal que $d_P = \sum_{i=0}^{N-1} d\left[c_i, c_{i+1 mod(N)}\right]$ sea mínimo. La distancia entre cada ciudad viene dada por la matriz $D : NxN$, donde $d[x, y]$ representa la distancia que hay entre la ciudad X y la ciudad Y.

La solución más directa es la que aplica la fuerza bruta: evaluar todas las posibles combinaciones de recorridos y quedarse con aquella cuyo trazado utiliza la menor distancia. El problema reside en el número de posibles combinaciones que viene dado por el factorial del número de ciudades ($N!$) y esto hace que la solución por fuerza bruta sea impracticable para valores de N incluso moderados con los medios computacionales actualmente a nuestro alcance.

Por ejemplo, si un ordenador fuese capaz de calcular la longitud de cada combinación en un microsegundo, tardaría algo más 3 segundos en resolver el problema para 10 ciudades, algo más de medio minuto en resolver el problema para 11 ciudades y. . . 77,146 años en resolver el problema para sólo 20 ciudades. Por ejemplo las rutas posibles entre 12 ciudades son (479 millones) 479,001,600 combinaciones y los caminos individuales entre ciudades son el sumatorio de las $12 - 1$ ciudades es decir 66.

1.9. Modelos amenazadores

Debemos considerar un modelo fuerte de amenaza para los protocolos de votación. Consideramos amenazas a partir de tres fuentes separadas: DREs, totalidad, y partidos coactivos del exterior. Para empeorar las cosas, los partidos corruptos pueden corromper juntos. Por ejemplo, DREs manipulados pueden coludir con coactivos

exteriores para comprar votos. Un programador en la fabricación podía insertar un código Troya, o al velador en la estación de interrogatorio podría instalar un código la noche antes de la elección. Debemos asumir que DREs manipulados se comportan arbitrariamente. Verificar el software de los DRE en una elección es muy complicado, y una meta de los esquemas de Neff y Chaum son eliminar la necesidad de verificar que el software de DRE esté libre de caballos de Troya. También debemos considerar partidos corruptos en marcar procesos, tal como manipular el pizarrón o funcionarios corruptos. Estos partidos manejan un poder significativo, y pueden causar grandes problemas si son corruptos. Por ejemplo, si el pizarrón es manipulado, puede borrar todas las boletas. Si todos los software usados por los funcionarios es manipulado, podría borrar a las porciones privadas de las llaves de los funcionarios, haciendo el desciframiento de las boletas imposible. Para evaluar la resistencia de coerción de un sistema de votación, se debe considerar coactivos exteriores coludiendo con votantes corruptos. Asumimos que el coercer no está presente en la cabina de votación. Los ataques donde la coerción esta físicamente presente son fuera del alcance de los protocolos de votación y pueden ser contenidos solamente con mecanismos físicos de seguridad. Similarmente, ataques donde un votante registra sus acciones en la cabina de encuestas (es decir., con una cámara de video o cámara del celular) está también fuera del alcance de los protocolos de votación. Finalmente, debemos considerar a votantes honestos pero no fiables. Por ejemplo, los votantes y los trabajadores de las encuestas no pudieron entender completamente la tecnología de votación o utilizar la característica de verificar, y un partido corrupto tomo ventaja de esta ignorancia, apatía, o fallo para afectar el resultado de la elección.

Capítulo 2

Protocolos criptográficos

Este capítulo describe ampliamente lo que es un protocolo critográfico, mencionando los tipos de protocolos que existen, al igual que la explicación de algunos protocolos en la historia de la humanidad y algunos la forma en como se realizá el protocolo, asi como su forma de ser usado. Por otra parte se mencionan dos protocolos, el primero que ayuda a la seguridad del sistema en la votación sobre todo a que los mensajes pasen seguros por los canales usados, el critosistema ElGamal el cual se describirá, se explicará en este capitulo, otro protocolo que nos ayuda sobre todo para saber si una persona se dice ser quien es, mediante la prueba de Conocimiento-cero, se manejarán la forma de realizar la prueba, se explicará como funciona, caracteristicas a tener en cuenta sobre esta prueba.

Mas formalmente se darán los algoritmos a utilizar con su respectivo proceso ya sea para cifrar o darle seguridad a la votación o verificar la fiabilidad de los mensajes, claves, segun sea el caso que se requiera.

Existe una amplia variedad de protocolos criptográficos, que dan respuesta a diferentes objetivos. Algunos de ellos son:

Protocolos de Autentificación de Usuario Permiten garantizar que el remitente de un mensaje o el usuario con el que establecemos comunicación es realmente quién pretende ser.

Protocolos de Autentificación del Mensaje Garantizan que el mensaje enviado no ha sido substituido por otro ni alterado.

Distribución de claves Un problema importante en Criptografía de clave privada es el de la creación y transporte de las claves a utilizar por cada par de usuarios. En cuanto a las claves de un sistema de clave pública, la problemática de su distribución es distinta, demandando protocolos específicos.

Protocolos para Compartir Secretos Su objetivo es distribuir un cierto secreto (por ejemplo la clave para abrir una caja fuerte), entre un conjunto X de participantes, de forma que ciertos subconjuntos prefijados de X puedan, uniendo sus participaciones, recuperar dicho secreto.

Pruebas de Conocimiento-Cero Permiten a un individuo convencer a otro de que posee una cierta información, sin revelarle nada sobre el contenido de la misma.

Transacciones Electrónicas Seguras Permiten realizar de manera electrónica segura las operaciones bancarias habituales, firma electrónica de contratos, etc.

Compromiso de bit Permiten a una parte A comprometerse con una elección (un bit o más generalmente una serie de bits) sin revelar tal elección hasta un momento posterior. El protocolo garantiza a otra parte B que A no modifique su elección.

Transferencias Transcordadas Permiten a una parte A enviar a otra B un mensaje o secreto entre dos posibles. A no conoce cual de los dos ha recibido realmente B.

Elecciones Electrónicas Permiten realizar un proceso electoral electrónicamente, garantizando la deseable privacidad de cada votante y la imposibilidad de fraude.

Jugar al Poker por Internet Posibilita a dos personas, físicamente separadas, mantener una partida de poker (o similar: cara o cruz, chinos, etc), comunicándose por correo electrónico, teléfono, etc, garantizando la imposibilidad de hacer trampa.

DeMillo y Merritt [**?**] proponen un protocolo de votación electrónico basado en multi cifrados en el voto del votante usando dos claves públicas de cada uno de los otros votantes. Sus protocolos requieren la cooperación de todos los votantes registrados. Si cualquier votante se abstiene o decide no cooperar el esquema entero falla.

Recuperar el protocolo entero tiene que ser reiniciado. Así este esquema, aunque asegura la privacidad del votante, es conveniente para "votación en salas de reunión" escenarios y no para la votación en gran escala.

Los esquemas propuestos por Cohen y Fischer[**?**], Benaloh y Young[**?**], y Cramer y otros [**?**] son similares los unos de los otros. Utilizan técnicas del Conocimiento-Cero complejo para alcanzar el anonimato del votante. Los protocolos de Cohen y Fischer[**?**], Benaloh y Young [**?**] se basan en alto grado de cifrado residual. Además, el protocolo de Cohen y Fischer [**?**] proporciona anonimato al votante asumiendo que

se confía en los funcionarios de casilla.

El protocolo Benaloh y Young [**?**] mejora sobre el esquema Cohen y Fischer [**?**] distribuyendo las porciones del excedente de las boletas a un número de autoridades que las marcan. Esta extensión ofrece más protección a la privacidad. Sin embargo, el problema de ambos es que implican cómputos complejos y de alto grado que no pueden ser fácilmente disponibles.

La aproximación propuesta por Cramer y otros [**?**] mejora la eficiencia del protocolo del Conocimiento-Cero usando el cifrado de logaritmo discreto. Una boleta que será emitida por un votante se considera un secreto. El votante envía una parte comprobable de la boleta secreta a cada autoridad que compilara los votos. Cada autoridad comprueba la validez de la parte recibida.

Las autoridades cuentan los votos independientemente y envían los resultados parciales a la autoridad central. Este último junta los resultados parciales para conseguir el resultado final. Cada autoridad que compiló los votos consigue solamente una sola pieza de un voto y no sabe nada sobre el votante mismo.

La autoridad central consigue solamente resultados agregados de las autoridades que compilaron los votos y no pueden saber los votos, solamente los resultados. Así la privacidad de los votantes se asegura. El problema de esta aproximación es el cómputo implicado en el esquema y la comunicación implicada. Un segundo problema es que los votantes solo pueden votar si/no a una pregunta fija.

Iversen [**?**] propone un esquema basado en el concepto de homomorfismo privado probabilístico y requiere la comunicación entre los votantes y candidatos. El esquema no requiere la cooperación entre los votantes y asegura que ningún grupo de votantes pueda interrumpir la votación. Preserva la privacidad de los votantes contra cualquier grupo de candidatos, de funcionarios y de votantes deshonestos.

Sin embargo, el esquema es computacionalmente intenso y la comunicación es estrictamente prohibida cuando el número de votantes es grande. Así el esquema no es conveniente para la elección en gran escala por Internet. Por otra parte el protocolo no es conveniente para los sondeos de opinión donde no existe el concepto de una entidad similar a un candidato.

Nurmi y otros[**?**]. Proponen dos protocolos el protocolo de dos agencias y el pro-

tocolo de una agencia. En el primero un certificador se asegura de que solamente los votantes elegibles puedan votar y un compilador de votos recopile los votos y publique los resultados. El certificador proporciona etiquetas de identificación a cada votante. El votante vota con las etiquetas y no con su identificación.

El problema con este protocolo es que el certificador puede acceder a un voto otra vez y regresarlo al votante y, si el compilador de votos colude con el certificador, la privacidad del votante se compromete. También si algunos votantes no votan, el compilador de votos y el certificador pueden coludir para emitir votos falsos.

En el protocolo de una agencia el compilador de votos el mismo distribuye las etiquetas usando un protocolo "revelador de secretos todo o nada". El protocolo es complejo y requiere la comunicación de los votantes entre ellos mismos. Esto soluciona el problema de la colusión en el protocolo de dos agentes. Sin embargo, no soluciona el problema del compilador de votos que emite votos falsos contra esos votantes que reciban etiquetas pero deciden no emitir su voto.

El protocolo propuesto por Fujioka y otros[**?**]. Utiliza "firmas ciegas" y los canales anónimos para asegurar la privacidad del votante. En este protocolo el votante prepara una boleta llena, la cifra con una clave secreta, y la firma ciega usando un número al azar. El votante después de firmar la boleta con una firma ciega la envía a un certificador.

El certificador primero verifica que la firma pertenezca a un votante registrado que todavía no ha votado, después lo firma y se lo devuelve al votante. El votante quita el factor que ciega para conseguir la boleta que esta firmada por el certificador.

El votante entonces envía la boleta cifrada firmada a un compilador de votos vía un canal anónimo. El compilador de votos comprueba la firma en la boleta cifrada. Si la boleta es válida, el compilador de votos la pone en una lista que se publica al final de la votación.

Los votantes pueden verificar que sus boletas están en la lista. Una vez que se verifica, los votantes envían al compilador de votos las claves del desciframiento necesarias para abrir sus boletas, otra vez vía los canales anónimos.

El compilador de votos utiliza estas claves para descifrar las boletas y agregar los votos a la elección. Después de la elección el compilador de votos publica las claves

descifradas junto con las boletas cifradas de modo que los votantes independientemente puedan verificar los resultados de la elección.

Fujioka y otros [**?**] mencionan que usando canales anónimos aseguran la privacidad del votante. Sin embargo, la boleta cifrada que es emitida por el votante contiene la firma del votante que permite que el votante identifique su boleta en la lista publicada.

Así cualquier entidad que pueda verificar la firma del votante puede ligar un votante a una boleta emitida y el anonimato del votante no es segura. Una segunda desventaja importante es que todos los votantes registrados deben emitir sus votos y ningún votante puede abstenerse.

Si un votante se abstiene entonces el certificador y el compilador de votos pueden coludir para emitir votos falsos en el final. Estos votos falsos no pueden ser detectados. Un tercer problema del protocolo es que cada votante necesita estar implicado en el proceso de votación hasta el final de la fase de votación. Esto es porque un votante no puede introducir su clave descifrada hasta después de la fase de votación.

El protocolo de Sensus [**?**] se basa en el protocolo propuesto por Fujioka y otros[**?**]. El protocolo de Sensus [**?**] supera los dos problemas pasados del protocolo de Fujioka[**?**]: el votante puede introducir la clave descifrada inmediatamente después de que acepta el recibo del compilador de votos sobre la boleta cifrada; si un votante decide abstenerse, el votante lo indica en el voto.

Sin embargo, si el votante se comporta irresponsablemente y decide regresar la boleta cifrada, es decir no hace ninguna indicación en su boleta, entonces este conocimiento se puede explotar por colusión del certificador y del compilador de votos para emitir boletas falsas.

Park y otros[**?**]. Utiliza los canales anónimos para emitir votos. Pfitzmann [**?**] ha demostrado posibles esquemas de cómo romper el anonimato del esquema de Park y otros[**?**].

Juang y Lei [**?**] propone un protocolo seguro de votación electrónico que utilice los criptosistemas del umbral para preservar la imparcialidad de las campañas de los candidatos y la capacidad del votante de verificar su voto en la cuenta final. El protocolo propuesto es complejo en términos de cómputos criptográficos y requiere

los canales anónimos para emitir votos.

Okamoto [**?**, **?**] propone dos protocolos con el objetivo de prevenir la compra de votos. Estos protocolos aseguran que después de que un votante emite su voto, el votante no posee ninguna evidencia para probar que él votó de cierta manera - una característica conocida como recibo-libre.

El protocolo utiliza canales anónimos para emitir un voto. Sin embargo el protocolo sufre de la desventaja que un votante puede ser forzado por un candidato para votar de cierta manera.

Entre estos protocolos uno requiere la existencia de un canal inexplotable. El otro requiere la existencia de una cabina de votación pero siendo un lugar físico en el cual el votante pueda interactuar y comunicarse secretamente con una segunda entidad.

Esto es una fuerte suposición y puede no ser aceptable Andrew Neff [**?**, **?**] y David Chaum [Cha85] propusieron esquemas revolucionarios para DRE(Grabación electrónica directa) basada en votación electrónica. La pieza central de estos esquemas consiste en un nuevo y sofisticado protocolo criptográfico que permite que los votantes verifiquen si sus votos son contados correctamente.

Compañías de votación Votegrity y Vote. Han implementado los esquemas de Chaum [Cha85] y de Neff[**?**, **?**], respectivamente. Estos esquemas representan un significativo avance sobre los previos DRE basados en votación electrónica: los votantes pueden verificar que sus votos se hayan registrado correctamente, y cada uno puede verificar que el procedimiento que marca esté correcto, preservando privacidad y coerción en el proceso.

En el análisis de estos esquemas criptográficos, se encontró dos debilidades: canales subliminales en las boletas cifradas y los problemas resultado de falta de fiabilidad humana en protocolos criptográficos. Estos ataques podrían potencialmente comprometer la integridad de la elección, evita la privacidad del votante, y permite la coerción del voto. Se encuentran perceptibles pero irrecuperables negaciones de ataques de servidores.

Estas debilidades llegaron a ser solamente evidentes al examinar el sistema en su totalidad, subrayando la importancia de un análisis de seguridad que observa los protocolos criptográficos en su contexto de sistemas.

La verdadera prueba para estas debilidades depende de cómo estos esquemas finalmente se ponen en ejecución. David Chaum [Cha85] y Andrew Neff [**?**, **?**] cada uno ha propuesto un protocolo de votación criptográfico para el uso en máquinas de DRE. Aunque estos protocolos difieren en los detalles de su operación, son estructuralmente similares.

Ambos protocolos consisten en cuatro etapas: inicialización de la elección, preparación de la boleta, conteo de boletas, y verificación de la elección. Antes de la elección, seleccionamos funcionarios de casilla competentes, elegidos tal que es inverosímil que se corromperán todos ellos.

Durante la inicialización de la elección, los funcionarios interactúan entre si antes de la elección para elegir parámetros y para producir el material que se usara durante el protocolo. Los funcionarios representaran un amplio conjunto de intereses grupales y agencias gubernamentales para garantizar la suficiente separación del privilegio y para desalentar la corrupción entre los funcionarios.

La preparación de las boletas comienza cuando un votante visita una sala de interrogatorios para emitir su voto el día de la elección, y termina cuando se emite en la boleta. Para emitir su voto, el votante interactúa con un DRE en privado en una cabina de votación para seleccionar sus opciones de la boleta.

El DRE entonces produce una boleta electrónica que representa la opción del votante y lo postea a un pizarrón de boletas públicas. Este pizarrón sirve como la urna. Al mismo tiempo, el DRE interactúa con el votante para proporcionar un recibo.

Los recibos se diseñan para resistir la compra de votos y la coerción, y no permiten que el votante pruebe a terceros cómo votó. También, a la boleta de cada votante se le asigna un número de serie único de boleta (BSN). $BSNs$ fácil en la revisión y verificación de procedimientos sin comprometer la privacidad de la votación.

Después de que todas las boletas se hayan anunciadas en el pizarrón, la etapa del conteo de boletas comienza. En el conteo de boletas, los funcionarios ejecutan una red-mix con multipasos verificable públicamente, donde cada funcionario en privado ejecuta una etapa en particular de la red-mix. Para mantener el anonimato, los funcionarios sacan cada boleta de los BSN antes de que entren en la red-mix.

Cada etapa de la red-mix toma como entrada un conjunto de boletas cifradas, las descifran parcialmente o recifran parcialmente (dependiendo del estilo de la red-mix), y las permuta aleatoriamente. El resultado final de la red-mix es un conjunto de claves de boletas que pueden ser contadas públicamente pero que no puede ser ligado a las boletas cifradas o a la identidad del votante.

En protocolos de votación criptográficos, la red-mix se diseña para ser universalmente comprobable: el funcionario proporciona una prueba que cualquier observador pueda utilizar para confirmar que el protocolo se ha seguido correctamente. Esto significa que un funcionario corrupto no puede agregar, suprimir, o alterar boletas. En los puntos de este proceso, los votantes y los observadores pueden engancharse a la verificación de la elección.

Después de que su boleta ha sido registrada en el pizarrón, el votante puede utilizar su recibo para verificar que su voto fue según lo previsto y será avalado en el resultado de la elección. El recibo no sirve como expediente oficial de la selección del votante; es solo un intento para convencer al votante que su boleta fue registrada correctamente.

Observadores de la elección puede verificar ciertas características sobre boletas en el pizarrón, por ejemplo, que todas las boletas son bien formadas o que el procedimiento de la red-mix fue realizado correctamente. Los protocolos de Chaum [Cha85] y de Neff [**?**, **?**] requieren DREs para contener los dispositivos de impresión especiales para proporcionar los recibos. Los requisitos de seguridad para la impresora son:

- el votante puede examinar su salida, y
- ni el *DRE* ni la impresora puede borrar, cambiar, o sobrescribir cualquier cosa ya impresa sin que el votante lo detecte inmediatamente. Hay algunas diferencias en las pruebas que se realizan en adición a los requerimientos de seguridad que deben conocer.

Los esquemas de votación de Neff [**?**, **?**] y Chaum [Cha85] se esfuerzan en alcanzar las metas siguientes:

Emitir-intento La boleta de un votante en el pizarrón debe representar exactamente sus opciones.

Contar-emitir La cuenta final debe ser una cuenta exacta de las boletas en el pizarrón.

Comprobable Las dos características anteriores deben ser comprobables. Comprobar Emitir-intento significa que cada votante debe ser capaz de verificar su boleta en el pizarrón representa exactamente el voto que hizo. Comprobar Contar-emitir significa que cada uno puede verificar que la cuenta final es una cuenta exacta de las boletas contenidas en el pizarrón.

Un votante/un voto Las boletas en el pizarrón debe representar exactamente los votos emitidos por votantes legítimos. Los partidos corruptos no podrán agregar, duplicar, o cancelar boletas.

Resistencia a la coerción Un votante no será capaz de probar cómo votó a terceros no presentes en las votaciones.

Privacidad Las boletas deben ser secretas.

2.1. Criptosistema ElGamal

Esta sección describe una forma de dar seguridad al envio de los mesajes mediante canales inseguros, en este caso refiriendonos al sistema ElGamal se explicará como se genera, su forma de comprobar a los usuarios y estos mismos comprobar en el sistema , el sistema empleado para generarlo y algunas características que se deben tomar en cuenta sobre todo para armar un sistema de este tipo. Sabiendo la forma establecida de formar el sistema, se establece un algoritmo general para firmar los mensajes mediante el uso de ElGamal.

ElGamal esta buscuando la mayor seguridad para los mensajes, es decir, que lo que ofrece este sistema es una forma segura de enviar mensajes a traves de canales inseguros y que sean dificiles de conocer por eso mediante el uso de ElGamal se dara seguridad a los mensajes que se requieren de los votantes en este caso para que se tenga la elección segura.

En el procedimiento de firmado Elgamal no se firmara el mensaje con el procedimiento RSA atraves del orden cifrado y descifrado sino atraves de una operación mas compleja. Esto trae como consecuencia que la firma del mensaje no se puede firmar de regreso. Para generación y verificación de una firma digital se usaran en el sistema de firma Elgamal la clave privada t y la clave pública $\tau = g^t \mod p$ para la generación de firma de un mensaje el participante T va necesitar seguir estos pasos:

1. escoge un numero r aleatoriamente de las $p-1$ opciones que se tiene y forma

$$k \equiv g^r \mod p$$

2. calcula una solución s a la congruencia

$$tk + rs \equiv m \mod p-1$$

puede haber pasado por ejemplo utilizando el algoritmo de Euclides se puede encontrar el inverso multiplicativo r^{-1} de r en $\mod p$ del conjunto z^p obteniendo

$$s \equiv (m - tk)r^{-1} \mod p-1$$

Por lo tanto la firma se obtiene de la pareja (k, s).

El receptor del mensaje firmado $(m, (k, s))$ puede probar la firma mediante la obtención de los valores $g^m \mod p$ y $\tau^k k^s \mod p$ y lo compara con los mandados por el firmante y verifica si son identicos. El esquema de firmado Elgamal es viable ya que por el teorema de Fermat se cumple la igualdad

$$g^m \equiv g^{tk+rs} \equiv \tau^k k^s \mod p$$

para cada documento que este correctamente firmado $(m, (k, s))$.

Tambien es seguro ya que hasta la fecha no se ha encontrado ningun procedimiento de ataque eficiente a este algoritmo.

Al contrario del algoritmo RSA en forma original no se puede obtener ganancia de la firma del mensaje. Una variante especialmente eficiente del procedimiento Elgamal una idea de C. Schnorr [Sch90], el cual fue usado en los Estados Unidos para la norma de generación de firmas digitales bajo el nombre *Digital Signature Standart*.

El esquema de firma ElGamal puede semejar a la clave de acuerdos de Diffie-Hellman en el modo transferencia de clave. Su seguridad se basa en la dificultad de resolver el problema del logaritmo discreto y del problema de Diffie-Hellman.

El esquema de firma ElGamal genera firmas digitales con apéndice sobre mensajes binarios de tamaño arbitrario, y requiere una función hash (h) para obtener el sumario de los mensajes a firmar.

Generación Cada entidad crea una clave pública y su correspondiente clave privada.

Cada entidad A debe hacer:

1. Genera al azar un número primo grande p y un generador α del grupo multiplicativo Z_p^* de los enteros modulo p
2. Seleccionar al azar un número entero a, $1 \leq a \leq p-2$, y calcular $\alpha^a \mod p$
3. La clave pública de A es $(p; \alpha; \alpha^a)$. Una clave privada de A es a.

Cifrado B cifra un mensaje m para A, que A descifra. Para cifrar B debe:

1. Obtener la clave pública auténtica de A (p, α, α^a).
2. Representar el mensaje como un número entero m en el rango $\{0, 1, \ldots, p-1\}$.
3. Selecciona al azar un número entero k, $1 \leq k \leq p-2$.
4. Cálcular
$$\gamma = \alpha^k \mod p$$
y
$$\delta = m(\alpha^a)^k \mod p$$
.
5. Enviar el texto cifrado $c = (\gamma, \delta)$ a A.

Descifrado Para recuperar el mensaje llano m de c, A debe:

1. Utilizar la clave privada a para calcular
$$\gamma^{p-1-a} \mod p$$
(nota: $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$).
2. Recuperar m calculando $\gamma^{-a}\delta \mod p$.

Ejemplo

De cifrado ElGamal con parámetros pequeños

- *Generación de clave* La entidad A selecciona el primo $p = 2357$ y un generador $\alpha = 2$ de Z_{2357}^*. A elige la clave privada $a = 1751$ y calcula
$$\alpha^a \mod p = 2^{1751} \mod 2357 = 1185.$$
Una clave pública de A es $(p = 2357; \alpha = 2; \alpha^a = 1185)$.

- *Cifrado* Para cifrar un mensaje $m = 2035$, B selecciona al azar un número entero $k = 1520$ y calcula

$$\gamma = 2^{1520} \mod 2357 = 1430$$

y

$$\delta = (2035)(1185^{1520}) \mod 2357 = 697.$$

B envía $\gamma = 1430$ y $\delta = 697$ a A.

- *Descifrar* Para descifrar A calcula:

$$\gamma^{(p-1-a)} = 1430^{605} \mod 2357 = 872,$$

y recupera m calculando

$$m = (872)(697) \mod 2357 = 2035.$$

Parámetros comunes de un sistema ancho (system-wide) todas las entidades pueden utilizar el mismo número primo p y el generador α, en este caso p y α no necesitan ser publicados como parte de la clave pública.
Esto resulta en claves públicas de menor tamaño.

Eficiencia el cifrado ElGamal las siguientes características

1. El proceso de cifrado requiere dos exponenciaciones modulares, llamados $\alpha^k \mod p$ y $(\alpha^a)^k \mod p$. Estas operaciones pueden acelerarse seleccionando al azar exponentes k teniendo alguna estructura adicional.
2. Una desventaja del cifrado ElGamal es que hay una expansión de mensaje por un factor de 2. Es decir, el texto cifrado es dos veces mas largo que el mensaje llano correspondiente.

Cifrado al azar El cifrado ElGamal es uno de los tantos esquemas de cifrado que utilizan el azar en el proceso de cifrado otros incluyen el cifrado de McEliece, y Goldwasser-Micali, y Blum-Goldwasser cifrado probabilistico. Los esquemas de cifrado deterministas como RSA pueden también emplear el azar.Para evitar algunos ataques. La idea fundamental detrás de la técnica del cifrado aleatorio es utilizar el azar para aumentar la seguridad criptográfica de un proceso de cifrado con uno o más de los métodos siguientes:

1. Incremento del tamaño del espacio de mensaje del mensaje llano;
2. Disminuir la eficiencia o efectividad de los ataques al usar mapeo uno-a-muchos en el cifrado del mensaje llano; y
3. Disminuir la eficiencia o efectividad de ataques estadísticos nivelando la distribución de probabilidades a priori de entradas.

Seguridad El esquema de ElGamal tiene las siguientes características

1. El problema de romper el esquema de cifrado de ElGamal, es decir, recuperando m dando $p, \alpha, \alpha^a, \gamma$ y δ es equivalente a resolver el problema de Diffie-Hellman. De hecho, el esquema de cifrado de ElGamal se puede ver como una simple acotación del intercambio de claves de Diffie-Hellman para determinar la clave de sesión $\alpha^{(ak)}$, y después cifrar el mensaje por una multiplicación con esa clave de sesión.
   Por esta razón, la seguridad del esquema de cifrado de ElGamal se dice que esta basado en problemas del logaritmo discreto en Z_p^*, aunque tal equivalencia no se ha probado.
2. Es crucial que diversos números aleatorios enteros k se usen para cifrar mensajes diferentes.
   Suponer que la misma k es usada para cifrar dos mensajes m_1 y m_2 y el resultado del texto cifrado es (γ_1, δ_1) y (γ_2, δ_2). Entonces $\delta_1/\delta_2 = m_1/m_2$, y m_2 podría ser fácil calcular si se conoce m_1.

Tamaño de parámetros recomendado Dado el último progreso en el problema del logaritmo discreto en Z_p^*, un módulo p de 512 bits proporciona solamente seguridad marginal de ataque concernientes. En 1996, se recomendaba un módulo p de por lo menos 768 bits. Para la seguridad a largo plazo, se debe utilizar 1024 bits o módulos más grandes.
Para parámetros comunes de sistemas anchos aún para claves de gran tamaño pueden ser autorizados. Esto es porque la etapa dominante en el algoritmo del cálculo introductivo para los logaritmos discretos en Z_p^* es el precalculo de una base de datos de los factores del logaritmo base, siguiendo que los logaritmos individuales pueden calcularse rápidamente. Calculando la base de datos de los logaritmos para un módulo p en particular comprometerá el secreto de todas las claves privadas derivadas del uso de p.

En la tabla siguiente se observa el año, los bits necesarios para formar un arreglo de seguridad siendo simétrico, y RSA, al igual que para el sistema hash

Año	Simétrico	Asimétrico		Logaritmo Discreto		Curva Elíptica	Hash
		Optimista	Conservador	Clave	Grupo		
2016	79	1273	1392	158	1273	158	158
2017	80	1300	1435	159	1300	159	159
2018	80	1329	1478	160	1329	160	160
2019	81	1358	1523	162	1358	162	162
2020	82	1387	1568	163	1387	163	163

La tabla nos dice que para tener un sistema seguro, es decir, que no se pueda romper y siga vigente aún, por ejemplo en el año 2018 siga seguro él sistema se necesita tener mínimo 80 bits para hacer el sistema simétrico, al igual que para tratar de romperlo se necesitan 160 bits de capacidad hash.

2.1.1. Cifrado generalizado

En esta sección se mencionan algunas de las formas de cifrado ElGamal al igual que se describe ya en forma general una forma de cifrar, es decir un esquema que se utiliza como base para cumplir los requisitos explicados anteriormente.

El esquema de cifrado de ElGamal se describe típicamente dentro del grupo multiplicativo Z_p^*, pero se puede generalizar fácilmente para trabajar en cualquier grupo cíclico finito G.

Como con el cifrado de ElGamal, la seguridad del esquema generalizado de cifrado de ElGamal se basa en la dificultad para resolver problemas del logaritmo discreto en el grupo G. El grupo G se debe elegir cuidadosamente para satisfacer las condiciones siguientes:

Para la eficacia La operación del grupo en G debe ser relativamente fácil de aplicarse; y

Para la seguridad El problema de logaritmo discreto en G debe ser computacionalmente intratable.

Lo que sigue es una lista de los grupos que aparecen para conocer estos dos criterios, de los cuales los primeros dos han recibido la mayor atención.

1. El grupo multiplicativo Z_p^* de números enteros módulo de un primo P.
2. El grupo multiplicativo $F_{2^m}^*$ del campo finito F_{2^m} de característica dos.
3. El grupo multiplicativo F_q^* del campo finito F_q, donde $q = p^m$, p primo.
4. El grupo de las unidades Z_n^*, donde n es número entero compuesto.
5. El grupo de la clase de un campo de números cuadráticos imaginarios.

Los pasos para generar un algoritmo de cifrado son los siguientes:

Generación de clave Cada entidad crea una clave pública y su correspondiente clave privada. La entidad A debe hacer lo siguiente:

1. Selecciona un grupo aproiado cíclico G de orden n, con un generador α. (Se asume aquí que G se escribe multiplicativamente).
2. Selecciona aleatoriamente un número entero a, $1 \leq a \leq n-1$, y calcular el elemento del grupo α^a.
3. Una clave pública de A es (α, α^a), junto con una descripción de cómo multiplicar los elementos en G. La clave privada de A es a.

Cifrado B cifra un mensaje m para A, el que A descifra. B debe hacer lo siguiente:

a Obtener la clave pública auténtica (α, α^a) de A.

b Representar el mensaje como un elemento m del grupo G.

c Seleccionar al azar un número entero k, $1 \leq k \leq n-1$.

d Calcular $\gamma = \alpha^k$ y $\delta = m(\alpha^a)^k$.

e Enviar el texto cifrado $c = (\gamma, \delta)$ a A.

Descifrado Para recuperar el mensaje llano m de c, A debe hacer lo siguiente:

a Utilizar la clave privada a para calcular γ^a y entonces calcular $\gamma^{(-a)}$.

b Recuperar m calculando $\gamma^{(-a)}$.

Parámetros comunes de un sistema ancho Todas las entidades pueden elegir utilizar el mismo grupo cíclico G y generador α, en este caso α y la descripción de la multiplicación en G necesitan no ser publicados como parte de la clave pública.

Ejemplo

Cifrado de ElGamal usando el grupo multiplicativo de F_{2^m}, con parámetros artificiales pequeños

- *Generación de clave* Una entidad A selecciona el grupo G para ser un grupo multiplicativo del campo finito F_{2^4}, cuyos elementos son representados por los polinomios sobre F_2 de grado menor a 4, y donde la multiplicación está formado por el módulo del polinomio irreducible $f(x) = x^4 + x + 1$. Por conveniencia, un elemento del campo $a_3x^3 + a_2x^2 + a_1x + a_0$ es representada por la secuencia binaria $(a_3a_2a_1a_0)$. El grupo G tiene orden $n = 15$ y un generador es $\alpha = (0010)$. A elige la clave privada $a = 7$ y calcula $\alpha^a = \alpha^7 = (1011)$. Una clave pública de A es $\alpha^a = (1011)$ (junto con $\alpha = (0010)$) y el polinomio $f(x)$ que define la multiplicación en G, si estos parámetros no son comunes a todas las entidades).

- *Cifrar*. Para cifrar un mensaje $m = (1100)$, B selecciona aleatoriamente un número entero $k = 11$ y cálcula $\gamma = \alpha^{11} = (1110)$, $(\alpha^a)^{11} = (0100)$, y $\delta = m(\alpha^a)^{11} = (0101)$. B manda $\gamma = (1110)$ y $\delta = (0101)$ a A.

- *Descifrar*. Para descifrar, A calcula $\gamma^a = (0100)$, $(\gamma^a)^{(-1)} = (1101)$ y finalmente se recupera m por el cálculo $m = (\gamma^a)\delta = (1100)$.

2.2. Pruebas de Conocimiento-Cero

Esta sección se explicará detalladamente en lo que consiste la prueba de conocimiento-cero, las caracteristicas que debe tener, los pasos a seguir para hacer la prueba, el entendimiento de la prueba mediante un ejemplo muy bien explicado. Se describirá el algoritmo Fiat-Shamir y como hacer la prueba de conocimiento-cero mediante el uso de este algoritmo, dando paso a paso la forma de generar un algoritmo Fiat-Shamir describiendo las dos partes que lo componen al igual que una pequeña forma de comprobarlo.

Las pruebas de Conocimiento-Cero primero fueron concebidas en 1985 por Shafi Goldwasser y otros, en un bosquejo de "La complejidad del conocimiento de la iníteíraícción de sistema prueba" mientras que este papel no inventó la interacción de los Sistema Prueba.

En criptografía, una prueba del Conocimiento-Cero o un protocolo del Conocimiento-Cero es un método interactivo para que una parte pruebe a otro que la declaración de a (generalmente matemática) es verdad, sin revelar cualquier cosa

con excepción de la veracidad de la declaración. Una prueba del Conocimiento-Cero debe satisfacer tres características:

1. *Viabilidad* si la declaración es verdad, el verificador honesto (es decir uno, después del protocolo correctamente) será convencido de este hecho por una prueba honesta.

2. *Corrección* si la declaración es falsa, ninguna prueba de engaño puede convencer al verificador honesto de que sea verdad, excepto con una cierta probabilidad pequeña.

3. *Conocimiento-Cero* si la declaración es verdad, ningún verificador de engaño aprendera algo, con excepción de este hecho. Esto se formaliza demostrando que cada verificador de engaño tiene poco de simulador, que dado solamente la declaración que se probará (sin ningún acceso a la prueba), pueda producir una transcripción que aparente ser una interacción entre la prueba honesta y el verificador de engaño. Las primeros dos son características mas en general de pruebas de sistemas interactivos. El tercero hace la prueba Conocimiento-Cero.

   Cuando se usan los sistemas de Conocimiento-Cero se desea convercer a otros de que conocemos cierto secreto, pero sin delatar específicamente el secreto.

   Tales sistemas son idealmente adecuados como sistemas de identificación, en particular para la verificación de identidades de personas. Una persona A se puede identificar con otra persona B a través de la comprobación de cierto secreto. Se requiere que satisfaga tal sistema lo siguiente:

   - B no debe conocer de antemano el secreto de A, y
   - durante el proceso no debe adivinar el secreto.

   En este caso B no tendrá la posibilidad de identificarse ante terceras personas como A.

   El procedimiento de Conocimiento-Cero satisface esta condición de forma óptima: B puede convencerse, con el requerido nivel de seguridad, de la identidad de A, sin que en el camino obtenga *alguna* información; en particular no descubre el secreto. A esto se le conoce como **Propiedad del Conocimiento-Cero**.

Los protocolos de Conocimiento-Cero son, en la práctica extremadamente importantes (control de acceso electrónico), como teóricamente interesantes, ya que en ellos se utilizan métodos matemáticos no triviales y teorias complejas.

Ahora veremos un ejemplo: Es común etiquetar las dos partes en una prueba del Conocimiento-Cero como Liza (la provedora de la declaración) y José (el verificador de la declaración). En esta historia, Liza ha destapado la palabra secreta usada para abrir una puerta mágica en una cueva.

La cueva se forma como un círculo, con la entrada en un lado y la puerta mágica bloqueando el lado opuesto. Jose dice que él le pagará por el secreto, pero hasta que él este seguro que ella realmente lo sabe. Liza dice que ella le dirá el secreto, pero hasta que ella reciba el dinero. Idean un esquema por el cual Liza pueda probar que ella sabe la palabra sin decirla a Jose.

Primero, José espera fuera de la cueva mientras que Liza entra. Etiquetamos las trayectorias izquierdas y derechas de la entrada como A y B. Ella toma aleatoriamente la trayectoria A o B.

Entonces, José entra en la cueva y grita el nombre de la trayectoria que él quisiera que ella regresara, A o B, elegido al azar. Con tal de que ella realmente sepa la palabra mágica, esto es fácil: ella abre la puerta, si es necesario, vuelve a lo largo de la trayectoria deseada.

Observar que José no sabe por cuál saldrá. Sin embargo, suponga que ella no sabe la palabra. Entonces, ella puede volver solamente por la trayectoria nombrada si José da el nombre de la misma trayectoria que ella entró.

Puesto que José elige A o B al azar, ella tiene posibilidad del 50 % de adivinar correctamente. Si repiten este truco muchas veces, por decir 20 veces, su posibilidad de anticipar todas las peticiones José llega a ser pequeña, y José se convencerá de que ella sabe el secreto.

Se pregunta, ¿por qué Liza no solo toma una trayectoria sabida que la forze a través de la puerta, y hace que José espere en la entrada? Ciertamente, eso probará que Liza sabe la palabra secreta, pero también permite la posibilidad

de escuchar detras de las puertas.

Seleccionando al azar la trayectoria inicial que Liza toma y evitando que José la sepa, reduce las ocasiones que José puede seguir a Liza y aprender no solo que ella sepa la palabra secreta, pero ¿cuál es la palabra secreta realmente?. Esta parte del intercambio es importante para mantener la cantidad de información revelada a un mínimo.

2.2.1. El algoritmo Fiat-Shamir

El más conocido, y en la práctica el más importante procedimiento de Conocimiento-cero es el algoritmo de Fiat-shamir. La seguridad de este algoritmo se basa en el hecho de que en la práctica es casi imposible calcular raíces cuadradas en $\mathbb{Z}_n^*$ que es en lo que se basa este algoritmo.

El algoritmo de Fial-Shamir consta -como la mayoría de los algoritmos criptográficos- de dos fases, la fase de generación de clave y la fase de aplicación.

En la fase de *generación* A genera 2 números primos grandes p y q y forma el producto $n = pq$. El número n es público, mientras que p y q solo los puede conocer A. Después A elige un número s y construye $v := s^2 \mod n$.

El número s es el secreto individual de A (s es por secreto), mientras que con la ayuda de v (la marca de identificación) se puede verificar si una persona conoce o no el secreto. Esto significa en particular que s debe permanecer secreto, mientras que v debe ser público.

Uno se puede imaginar que n es una constante del sistema y que los números s y v debe ser dados, para cada parte, por una central. En la fase de *aplicación* A debe convencer a B de que conoce el secreto s. Para ello A y B siguen el protocolo:

- A elige aleatoriamente un elemento r de $\mathbb{Z}_n^*$ y lo eleva al cuadrado módulo n: $x := r^2 \mod n$. Después A manda a B el valor x.
- B elige aleatoriamente un bit b y lo manda a A.
- Si $b = 0$ A manda el valor $y := r$ a B
  Si $b = 1$ A manda el valor $y := rs \mod n$ a B

- B verifica esta respuesta. En caso de que $b = 0$ comprueba que $y^2 \mod n = x$. En el caso $b = 1$ prueba el si la igualdad $y^2 \mod n = xv \mod n$

También este protocolo cumple las condiciones para los procedimientos de Conocimiento-cero:

Viabilidad Cuando A conoce el secreto s podrá convencer a B, ya que en $\mathbb{Z}_n^*$ se cumple:

$$y^2 \equiv (rs)^2 \equiv r^2 s^{2b} \equiv r^2 v^b \equiv xv^b \mod n.$$

Corrección Una falsa A' puede contestar a lo más una de las dos preguntas, $b = 0$ o $b = 1$.

Si se pudiese contestar ambas preguntas (con y_0 o con y_1), entonces se poseerá una de las raíces de v: de $y_0^2 = x$ y de $y_1^2 = xv$ se sigue $(\frac{y_1}{y_0})^2 = v$, y con ello es $(\frac{y_1}{y_0})$ una raíz cuadrada de v módulo n. Se puede entonces engañar en una ronda con la probabilidad de $\frac{1}{2}$.

Por otro lado, A' puede engañar en una ronda mínima con probabilidad $\frac{1}{2}$. Cuando suponga que B le pregunta b, ahorá podrá preparar la respuesta como: Cuando se tenga $x := r^2 v^{-b} \mod n$ y $y = r$, así B durante la verificación no podrá obtener ninguna irregularidad. Como arriba A' en una ronda t podra convencer de que es A sólo con probabilidad $\frac{1}{2}^t$.

Conocimiento-Cero: Un simulador M puede realizar un diálogo con B de la siguiente manera:

- M elije aleatoriamente un bit c y un número r; después calcula $x := r^2 v^{-c} \mod n$ y manda x a B
- B responde con un bit b
- si $b = c$, entonces M manda el mensaje $y = r$ a B. En este caso si la verificación es exitosa se tendrá:

$$xv^b \equiv r^2 v^{-c} v^b \equiv r^2 \equiv y^2 \mod n$$

La terna (x, b, y) representa un paso de esta simulación de plática.

Si b es distinto de c entonces todos los mensajes enviados seran borrados y la simulación en esta ronda debe empezar de nuevo.
Tanto en el diálogo original, como en el previo aparecen el mismo número (aleatorios)

de ternas (x, b, y) para las cuales la igualdad $xv^b = y^2$ es válida. Ambos diálogos no pueden ser diferenciados por un observador.

El esquema de *Fiat-Shamir* es especialmente bueno para la comprobación de identidades de personas correspondientemente.

Capítulo 3

Protocolos criptográficos de votación

Este capítulo describe los protocolos más usados para la votación electrónica que son los usados con red-mix y los usados con cifrado homomorfo, y de estos se describirán abundantemente sobre lo que hacen, algunos tipos de protocolos que se ocupan, la forma en como hacerlos, la forma en como se usan.

Hay varias clases de protocolos propuestos para votación electrónica. Por ejemplo, en los protocolos basados en esquemas de firma ciega, el votante primero obtiene un símbolo, que ha sido firmado ciegamente por el administrador y que es solo conocido por el votante mismo. Posteriormente envía su voto anónimo, con este símbolo como prueba de legitimidad.

En esquemas que se usa el cifrado homomórfico, el votante coopera con el administrador para construir un cifrado del voto. El administrador explota las características homomórficas del algoritmo de cifrado para calcular la cuenta cifrada directamente de los votos cifrados. Aún en otros esquemas, los sistemas elaborados de redes-mix se emplean para garantizar la seguridad del votante.

3.1. Con Redes Mix

Esta sección se describe la forma del protocolo con red-mix, para despues utilizarlo en la votación como una forma de ayuda para la privacidad de los votantes. Se plantearán las formas de eliminar a los votantes deshonestos, así como la forma de comprobar que el sistema no este alterado. Mediante la red-mix que se maneja se

darán algunas formas de que la mix sea confiable en todo el proceso, es decir que se pueda verificar en cualquier momento, para ello se darán protocolos para revisar la forma en que se este llevando a cabo la votación.

Primero se elaboran los elementos para el sistema de votación, una vez que se tienen los elementos se pasara a explicar los tipos de red-mix conocidos para este sistema, después utilizar los tipos de red-mix para elaborar el sistema completo del decifrado y re-cifrado con sus respectivos pasos a seguir para cada uno de ellos.

Luego de explicar la forma de descifrar y re-cifrar este último se ocupara para dar un protocolo en especial utilizado con ElGamal, y este protocolo para cumplir ciertas características que después se manejan y se da a conocer como verificar que se cumplen.

Ya que se tiene el protocolo el usuario puede verificar si este, en el proceso, no esta alterado o dañado, por eso se maneja la forma de que el votante pueda verificarlo, pero también puede haber alteraciones del otro lado de la votación, es decir, del lado del votante por lo que por eso también se da a concer la forma de prevenir la corrupci ón por este lado, otro elemento que se maneja en la votación son los servidores por lo que también se daran ha conocer la forma de prevenir agravios por ellos.

Ya que se manejan en el sistema los mix también al igual que los demás se tendrán que analizar por si ocurre algún problema, para ello se analizan cada uno de los sistemas mix, para realizar el trabajo primero se daran la forma de verificar las mix por pares,después dar la forma general de como comprobar todas las mix.

Para verificar las mix hay varias formas de hacerlo, como se menciono par a par, por una prueba de conocimiento, mediante protocolos como los de Chaum-Pedersen, Neff, otra forma también es mediante series,

La red-mix es una importante base para establecer anonimato. Es utilizada por muchos esquemas de votación. La idea principal es proporcionar un mecanismo criptográfico que recibe un conjunto de mensajes y sale el mismo conjunto, pero en orden aleatorio. Esto se realiza de tal manera que nadie, solamente la red-mix, puede decir qué mensaje entrante corresponde a qué mensaje saliente.

Unos o más servidores pueden formar una red-mix. Generalmente, se utilizan los servidores múltiples para mejorar la seguridad. Aunque si algunos de los servidores revelan sus secretos, por ejemplo la permutación utilizada, a pesar de esto el anonimato todavía se conservaría .

Hay dos modelos de redes-mix: el modelo de Chaum y el modelo de re-cifrado.

En el modelo de Chaum, también conocido como descifrado red-mix, el mensaje

está cifrado con la clave pública de cada mix individual. Así, el cifrado más externo es hecho usando la primera clave pública del servidor y el más próximo, con la ultima clave del servidor. Mientras recibe el mensaje cifrado cada mix permuta el mensaje y quita el cifrado de salida respectivo.

En la red-mix de re-cifrado los mensajes se cifran con solo una clave pública. Como los mensajes cifrados aturden los servidores, cada servidor re-cifra los mensajes, usando la misma clave pública, pero una aleatoriedad y permutación diferente en el cifrado. La mix no conoce el texto llano del mensaje. En ambos modelos, se requiere una cierta confianza en los servidores de la red-mix.

3.1.1. Votación vía red-mix

En esta sección se describen algunos referencias con las que se debe inciar la votación mediante la red-mix, por eso el objetivo es revisar la existencia de métodos de votación que involucran criptografía. Por lo que primero se dan a conocer los elementos que se deben tener en el sistema de votación, para que una vez que se tengan los elementos se da a conocer los tipos de red-mix que se manejan, que se explican con detalle. Mediante la red-mix se podra verificar y saber mucho con respecto a los votantes y su autenticidad a si como el buen desempeño del sistema

El paradigma de la red-mix

Se daran a conocer los elementos que se deben tener en el sistema, así como también explicación de los tipos de red-mix utilizados. Chaum fue el que inicialmente utilizo las redes-mix para votación, y desde entonces se ha utilizado en otros esquemas. El proceso es el siguiente:

1. n votantes crean n boletas, $B_1, B_2, \ldots, B_n$.
2. Cada votante cifra su boleta, completando el nivel 0 de textos cifrados: $C_{1,0}, C_{2,0}$, $\ldots C_{n,0}$.
3. Se tiene t mix, o servidores mix, $S_1, S_2, \ldots S_t$
4. Las i mix S_i tomadas de $\langle C_{1,(i-1)}, C_{2,(i-1)}, \ldots C_{n,(i-1)} \rangle$, secretamente permutan su orden y aun:
   - Recifra (recifra mix), o
   - Decrifra (descifra o Chaum mix),

Para obtener $\langle C_{1,i}, C_{2,i}, \ldots C_{n,i} \rangle$

5. La secuencia final de texto cifrado $\langle C_{1,t}, C_{2,t}, \ldots C_{n,t} \langle$ tal vez necesite un ciclo de descifrado (para un re-cifrado en malla).

6. Todas las salidas de las redes-mix son publicadas en pizarrones legibles e universalmente legibles, incluido el final total de las boletas de los textos llanos.

Confiando en el cifrado inicial Se define $Ci, 0 = Enc(B_i)$, y se quiere una prueba de conocimiento para B_i. Esto no es una verificación de votación porque el proceso de cifrado es elaborado por una maquina, y un humano no puede verificar que $C_{i,0}$, el cambio de voto, la prueba de votación , es en efecto el cifrado de B_i , la intención del votante.

Confiando en los servidores de operación ¿Se puede confiar en los servidores para las propiedades mix? Un honesto S_j puede dar seguridad, pero un deshonesto S'_j puede reemplazarlo, alterarlo, duplicar el voto, aun si todos los textos cifrados son publicados.

Tenemos dos tipos de red-mix: malla descifrada y malla re-cifrada.

Malla descifrado En una malla descifrado, cada servidor mix pela las capas de una cebolla de cifrados múltiple de un texto llano. Para el final, todas las capas son peladas y el texto llano listo. Más específico, cada servidor S_j tiene (PK_j, SK_j) para un esquema de cifrado con clave publica aleatoria (con una seguridad semántica). Cada texto llano es:

$$C_{i,0} = E(PK_1, E(PK_2, \ldots, E(PK_t, B_i)))$$

Como un esquema, todos los textos cifrados dados en capas deben tener el mismo tamaño, de otra manera son sencillos cruzarlos dando el servidor mix. Aleatoriamente (necesario para la seguridad semántica) significa que los textos cifrados se alargan con t (la aleatoriedad es descartada después de cada descifrado).

Malla re-cifrado Muchas mallas re-cifrado usan una variante del cifrado y re-cifrado ElGamal. En el cifrado ElGamal, se considera un grupo de enteros $\mathbb{Z}_p^*$ bajo la multiplicación y g, como generador de $\mathbb{Z}_p^*$. $x \in \mathbb{Z}_p^*$ es la clave secreta. $y = g^x$ es la clave pública. $E(m) = (g^r, (m)(y)^r)$ es la función cifrada aleatoria con $r \xrightarrow{R} \mathbb{Z}_{p-1}^*$ $D(c_1, c_2) = (c_1^x)^{-1} * c_2$ es la función descifrada en un par ElGamal. (El factor aleatorio es dividido fuera: tan largo como el par es un par

correcto ElGamal, descifrar es lo siguiente). Se describe el re-cifrado ElGamal (es decir, la re-aleatoriedad): $ReEnc(c_1, c_2) = (c_1 * g^s, c_2 * y^s) = (g_{r+s}, (m)(y^{r+s}))$ para $s \xrightarrow{R} \mathbb{Z}^*_{p-1}$. El re-cifrado no afecta el proceso de descifrado, ni requiere conocer la clave secreta.

La estructura general de las redes-mix se inicia con una fase de cifrado E, donde sus salidas aparecen en un pizarrón, en orden para asegurar la verificabilidad. Es seguida de muchas fases mix $mix_i, \ldots, mix_k$. Se requieren muchas fases para asegurar que sea un grupo grande.

En el descifrado de redes-mix, la fase mix revuelve y parcialmente descifra, cualquiera en las redes-mix de re-cifrado, la fase mix re-cifrado y revuelve. En las redes-mix de re-cifrado se anexa una fase final de descifrado D.

3.1.2. Descifrado Red-Mix

Esta sección describe como llevar a cabo un descifrado en una red-mix, esto nos permite verificar una red-mix los textos cifrados poderlos descifrar y volver a re-cifrarlos para no alterar los elementos originales.

Se explica con detalle los pasos a seguir para formar la fase de descifrado de red-mix que observandolo se entendera mejor, ya que el procedimiento se observo anteriormente.

Una red-mix de descifrado no tiene una fase de descifrado final. Mas bien, la fase inicial de cifrado E cifra las llegadas aplicando una serie k de operaciones de cifrado para cada llegada; cada mix pela una de estos cifrados aplicando su algoritmo de descifrado correspondiente; después de eso se mezcla todas sus llegadas descifradas aplicándoles una permutación aleatoria secreta.

Así, el esquema tiene la estructura de una cebolla; E construye la cebolla, y cada mix pela una capa de la cebolla. Mas especifico, cada mix tiene su propio par de claves. Se denota las claves de mix_i con (SK_i, PK_i). mix_i descifra sus llegadas usando sus claves (SK_i, PK_i); después de eso permuta secretamente todas sus llegadas descifradas.

El cifrado inicial E tiene las claves publicas de todas las mezclas $(PK_1, \ldots, PK_k)$; eso cifra cada llegada primero cifrando eso con PK_k, después cifra el resultado con PK_{k-1}, después cifra el resultado con PK_{k-2} y así. Así, si se denotan las boletas con

$B_1, \ldots, B_n$, entonces para cada $i = 1, \ldots, n$,

$$C_i = E(B_i) = E(PK_1 \ldots E(PK_{k-1}, E(PK_k, B_i)) \ldots).$$

Hay algunas salidas que necesitan ser direccionadas:

1. Los esquemas de cifrado seguros no esconden lo largo del texto llano, por eso se requiere que todos los textos llanos sean de la misma longitud.

2. El mix_k (el ultimo mix) genera la saliente final en la votación. Así, si la saliente no es la deseada se puede abortar. Una manera de prevenirlo es haciendo su segmento secreto, como una clave maestra.

3. La seguridad semántica esta cubierta, asumiendo que las boletas cifradas se publicarón solo después de que todos los votantes votarón. Por otro lado, se necesita tener una gran seguridad, como la $CCA2$.

3.1.3. Re-cifrado Red-mix

Esta sección describe la forma de usar el re-cifrado ya sea para construir nuevamente esquemas o re-contruir, o para verificar redes mix.

Una vez entendido el descifrado ahora se da la fase de re-cifrado por lo que sera más entendible, ya que es lo contrario a lo hecho en la fase anterior.

Lo opuesto a una fase mix en el descifrado red-mix, que mezclaba y descifraba parcialmente, el papel de una fase mix en el re-cifrado red-mix es solo mezclar. Desde que se revuelve, el conjunto resultante de los textos cifrados no cambia, y así para cada resultado de texto cifrado es fácil descubrir la conexión con el votante. Así una operación extra es necesario agregar a cada fase mix es la operación de re-cifrado.

En general, un re-cifrado red-mix consiste en una fase de cifrado inicial E, muchas fases mix $mix_i, \ldots, mix_k$, que revuelven y re-cifran, y una fase de descifrado final D. Tipicamente, el esquema de cifrado usado en el re-cifrado red-mix es el esquema de cifrado ElGamal.

ElGamal basado en el re-cifrado red-mix

Una vez conocido el re-cifrado se utilizara la base de este para realizar un esquema de cifrado ElGamal.

Marcando que en el esquema de cifrado ElGamal, uun cifrado de un mensaje m, con respecto a la clave publica (p, y, g), consiste del par $(g_r, (m)(y)^r)$, donde todas

las operaciones se hicierón modulo p, y $r \in_R \mathbb{Z}_q$ donde q es un primo grande que divide $p-1$, donde g es un generador del subgrupo de los elementos que en orden dividen q, y m esta en el subgrupo. La clave secreta corresponde a (p, g, y) es x tal que $g^x = y(\mod p)$.

El esquema de cifrado ElGamal tiene la propiedad para el re-cifrado que: cualquier mensaje cifrado $(a, b) = (g^r, (m)(y)^r)$ puede re-cifrarse escogiendo un número aleatorio $s \in_R \mathbb{Z}_q$ y calculando $(a)(g)^s, (b)(y)^s) = (g^{r+s}, (m)(y)^{r+s})$. Esta operación de re-cifrado resulta con un texto cifrado aleatorio para el mismo mensaje m.

Ahora se define ElGamal basado en el re-cifrado red mix :

1. Una clave publica ElGamal (p, g, y) es generada (de alguna manera distribuida).

2. La fase inicial de cifrado E cifra simple todas las boletas $B_1, \ldots, B_n$ aplicando el algoritmo de cifrado ElGamal con la clave pública $(p, g.y)$. Después de eso se dan a conocer los textos cifrados que resultan $(C_{1,0}, \ldots, C_{n,0})$ en el pizarrón.

3. Las i-mix fases, en la llegada un conjunto de textos cifrados $(C_{1,(i-1)}, \ldots, C_{n,(i-1)})$, re-cifran cada texto cifrado y permutan los textos cifrados resultantes usando una elección secreta de permutación aleatoria.

4. La fase de descifrado final D, dando un conjunto de texto cifrado $(C_{1,k}, \ldots, C_{n,k})$, descifran simple todos los textos cifrados de alguna manera distribuidos (en orden para asegurar un grupo grande).

Verificabilidad y Robustez

Las características que se deben cumplir en la votación vía mix se dan a conocer para dar una mayor seguridad al sistema.

Un sistema de votación se dice ser verificable si todos los votantes pueden verificar que sus votos fuerón contados. Un sistema de votación se dice ser robusto si un pequeño conjunto de servidores no puede alterar las elecciones. Los protocolos de red-mix no son verificables ni robustos.

Una característica que se requiere para cumplirse lo anterior es que cada servidor mix pruebe que contenga la operación correcta. Cada mix, requiere probar que existe una permutación π tal que $C_{j,i}$ es un re-cifrado de $C_{\pi(j)}, (i-1))$, para $j = 1, \ldots, n$.

Para que un texto cifrado sea un re-cifrado de otro: sean $c_1 = (\alpha_1, \beta_1) = (g^t, (m_1)(y^t))$ y $c_2 = (\alpha_2, \beta_2) = (g^u, (m_2)(y^u))$ cualesquiera dos textos cifrados. c_2 es un re-cifrado de c_1 si y solo si c_1 y c_2 son cifrados del mismo mensaje. Considerando la cuarteta.

$$\left(g, y, \frac{\alpha_2}{\alpha_1}, \frac{\beta_2}{\beta_1}\right) = \left(g, y, g^{u-t}, \frac{m_2}{m_1} y^{u-t}\right)$$

Así, c_2 es el re-cifrado de c_1 si y solo si $\left(g, y, \frac{\alpha_2}{\alpha_1}, \frac{\beta_2}{\beta_1}\right)$ es una cuarteta DDH. Es decir, cuarteta de la forma (g, y, g^r, y^r), que es equivalente a ser una cuarteta de la forma $(g, g^x, g^r, g^{(r)(x)})$. Así, probando que c_2 es el re-cifrado de c_1 transitivamente se probara que $(g, y, g^r, y^r) \in DDH$.

El protocolo Chaum-Pederson prueba que la cuarteta $(g, y, w, u) = (g, g^x, g^r, g^{(r)(x)})$ es una cuarteta DDH.

$$P \qquad\qquad V$$

$$s \in Z_q \xrightarrow[(a,b)=(g^s, y^s)]{} \qquad\qquad \xleftarrow{c} \quad c \in Z_q$$

$$\xrightarrow[t=s+cr]{} \text{acepta si y solo si}$$

$$g^t = (a)(w^c) \wedge y^t = (b)(u^c)$$

Es fácil verificar que el protocolo es un verificador honesto de conocimiento-cero de protocolos de prueba de conocimiento.

1. Neff propone una ligera diferencia en el re-cifrado red-mix, también basado en ElGamal. En el protocolo de Neff una operación de re-cifrado consiste parte en tomar un texto cifrado (a, b) y generar otro texto cifrado (a^c, b^c), para tomar un aleatorio $c \in_R \mathbb{Z}_q$. Esta operación cambia el mensaje cifrado m por m^c. La motivación dentro del esquema de Neff es que lo maneja para dar una prueba de conocimiento-cero eficiente, que envuelve solo lineal (en n) número de exponentes.

2. Hay protocolos rápidos que no son de conocimiento-cero, tal como el propuesto por Boneh y Golle y otro propuesto por Jacobsson, Jules y Rivest. Ambos usan tecnología nueva para verificarlo. En el primero, para cada servidor mix, el producto de un subconjunto aleatorio de sus llegadas es calculado, y el servidor mix se requiere para producir un subconjunto de salidas de productos iguales. En el segundo, llamado exploración parcial aleatorio, en el cual cada servidor da una evidencia fuerte de lo acertado de sus operaciones revelando un subconjunto seleccionado seudo aleaoriamente de sus relaciones de llegada y salida.

Una revisión al ElGamal basado en el re-cifrado red-mix

1. Votantes votan.
2. Una clave pública ElGamal (p, g, y) se hace (de una manera distribuida).
3. Se realiza la fase de cifrado inicial.
4. Todas las fases mix se realizan.
5. Cada fase mix produce una prueba. La prueba incluye una versión no interactiva de la prueba de Chaum-Pederson, obtenida aplicando el paso tipo Fiat-Shamir: el reto es calcular aplicando alguna función pseudo-aleatoria al primer mensaje y al contenido del pizarrón de las boletas publicadas; la clave para la función pseudo-aleatoria es escoger de una manera distribuida.
6. Todas las pruebas son revisadas, y si están correctas, entonces la fase de descifrado se realiza aplicando un umbral de descifrado. Si la prueba de la mix_i, falla, entonces el servidor erroneo mix_i, no es tomado en cuenta y todas las fases mix $mix_{i+1}, \ldots, mix_k$ se vuelven a efectuar.

3.1.4. El reto de la votación red-mix verificable

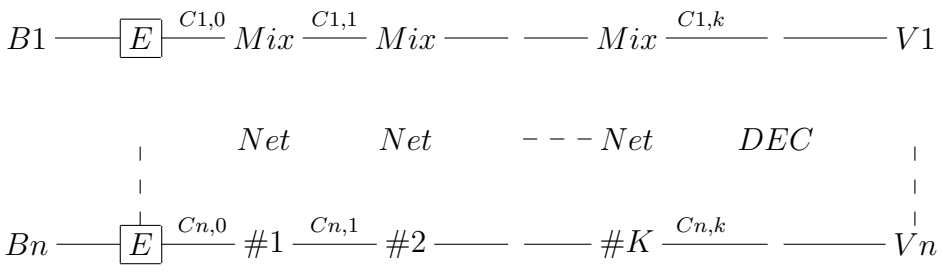
Si el votante quiere saber como esta funcionando el proceso y si no hay alteraciones esta sección describe la forma como el votante puede revisar el proceso electoral.

Ya que se conoce el protocolo de ElGamal usando re-cifrado entonces para que el votante pueda verificar que el proceso sea correcto se daran formas de como el votante pueda revisar los servidores y si no esten alterados o dañandos.

El manejo del cifrado ElGamal, es benéfico para re-cifrar cada texto cifrado con aleatorios nuevos sin conocimiento de lo que este contenido en las boletas. La salida de cada red-mix re-cifrado ElGamal pueden verificarse publicamente usando un

protocolo conocido para probar la equivalencia de dos logaritmos discretos. Un ciudadano conciente puede verificar la salida de un servidor de red-mix. Algunas formas de cómo un votante corrupto puede alterar el sistema y algunas técnicas para evitar esta situación.

Los pasos claves en ElGamal, para el protocolo de votación red-mix verificable:



Primero, un grupo desconfiados entre ellos mismos (como una fiesta republicana, una elección oficial, etc.) forman un protocolo de cálculo seguro para generar un par de clave ElGamal de la forma (g^x, x) donde la clave pública g^x es anunciada y la clave secreta x no es conocida, pero puede obtener de un umbral de los pedazos distribuidos entre las partes.

Las boletas $B_1, \ldots, B_n$ son generadas por n votantes diferentes y cifradas, bajo g^x, para los textos cifrados $C_{1,0} \ldots C_{n,0}$ con la ayuda de cálculos. Después los textos cifrados son enviados por los k servidores red-mix, operados por anfitriones con intereses loables. Todos los textos cifrados son publicados en un pizarrón. Una vez que las salidas de los servidores-mix son verificados, un umbral de las partes acuerdan descifrar las boletas y contar los votos.

3.1.5. Manteniendo al votante honesto

Lo primero que debemos revisar es que el votante sea el que debe de votar para eso esta sección explica la forma en como el votante puede alterar la votación ya sea en el texto cifrado o ser realmente el votante para ello se realizaría una prueba de conocimiento-cero, se describe como puede alterar el votante y como se puede comprobar que lo a alterado en algunos casos.

Al igual que se debe verificar si el sistema no este alterado, se debe prevenir que los votantes no intenten o traten de alterar también la votación por lo que se maneja modos de prevenir esto.

Cualquier protocolo de votación seguro debe defenderse contra grupos corruptos. El votante por si mismo debe poner atención en romper el sistema; esto es, llegar en diferentes caminos que aseguren la idea una votación democrática. Una por ejemplo, es la venta de votos.

Cualquier esquema de votación valido no debería permitir a una persona probar como voto a otra persona. Se observan otros dos problemas que pueden presentarse durante la iniciación de las boletas cifradas $C_{1,0} \dots C_{n,0}$.

Cancelando votos Suponga que a Carlos no le preocupan las eleccióones, pero desea frustrar el intento de Ana de expresar su opinióncualquiera que esta sea. Se dice que Carlos cancela el voto de Ana si, después de observar el voto cifrado de ella $C_{a,0}$ y sin conocimiento de la votación de ella, Carlos puede generar un texto cifrado $C_{b,0}$ con lo que vota de nuevo por Ana. Por ejemplo, si Ana vota que si en la salida 23, entonces se debe prevenir que Carlos realice una boleta cifrada con el voto opuesto (es decir, no en la salida 23) sin conocer por cual voto Ana.

Copy-cat Votación Otra situación que se debe prevenir es el duplicado de voto. Si Ana le dice a Carlos que ella votó que si en la salida 23 y el decide votar de la misma manera que ella lo hizo, eso esta bien. Pero si Ana decide no decirle a Carlos como votó ella, entonces el no será capaz de duplicar el voto. (No es difícil imaginar varios tipos de corrupción basados en el duplicado).

Los dos puntos anteriores se pueden prevenir si se pide a Carlos probar como votó; esto es, cuando Carlos presenta su boleta cifrada $C_{b,0}$ el debe estar preparado para dar una prueba de conocimiento-cero de conocer el texto llano contenido en la boleta.

Probando el conocimiento del texto llano

Al saber las formas de alterar la votación por parte del votante, es bueno hacerle una prueba al votante sobre como realizó su votación.

Dando un texto cifrado ElGamal $C = (\alpha, \beta) = (g^t, (m)(y^t))$ bajo la clave pública y, un votante puede probar conocer a m (es decir, como votó) probando que conoce t. La justificación es que el votante puede obtener m de C usando t. Un eficiente, protocolo de 3 rounds para probar el conocimiento de t, asume que g^t es pública.

$$P \qquad\qquad V$$

$$s \in Z_q \xrightarrow[A=g^s]{} \qquad \xleftarrow[]{c} \quad c \in Z_q \qquad \xrightarrow[r=s+ct]{} \text{revisar que:}$$

$$g^r = (A)(g^{t*c})$$

Aquí el probador P es el votante, posiblemente trabajando en conjunto con el software de cifrado E. Mientras que el verificador V puede ser un oficial de elección.

Verificador honesto de conocimiento-cero

Utilizando la prueba al votante, se puede realizar una prueba de conocimiento-cero.

El protocolo en la sección anterior es un verificador honesto de conocimiento-cero (HVZK) prueba del protocolo de conocimiento. Un protocolo HVZK tiene una propiedad especial: si el verificador ve dos transcripciones que usan el mismo conjunto inicial de la prueba $A = g^s$ donde se permite al verificador usar dos retos c_1, c_2 y recibir la respuesta $r_1 = s + (c_1)(t), r_2 = s + (c_2)(t)$, después el verificador calcula t. (Desde que el verificador conoce r_1, r_2, c_1, c_2, puede resolver para s y t).

3.1.6. Manteniendo la honestidad del servidor-mix

Ya que también se puede dar una alteración en el servidor esta sección explica la forma de como impedir una alteración en el sistema, y como comprobar que se ha llevado a cabo una mediante el uso del protocolo Chaum-Pedersen. Otra forma de verificar es revisando las mix utilizadas, se explicará el proceso a realizar.

Otra parte de la votación son los servidores por lo que también se manejaran formas de prevenir alteraciones por este lado.

Un servidor mix de re-cifrado se puede corromper de dos maneras: (1) puede intentar alterar el conteo de votos, y (2) puede intentar destruir la seguridad del votante. Para contrarrestar al primer adversario se debe forzar al servidor mix a probar que no ha alterado el conteo de votos.

Destruyendo la seguridad del votante, filtrando información acerca de la permutación usada, esto llega mas allá de las habilidades matemáticas para resolverlo. Esto es imposible para el oficial de elecciones verificar que un servidor mix no ha vendido su información a los malos.

Así, la efectividad del conteo de votos es segura, mientras que la seguridad de los votantes depende de que al menos uno de los servidores mix sea honesto (es decir, guardando la información secreta).

Cada servidor mix toma en k el texto cifrado ElGamal $C_i = (\alpha_i, \beta_i)$, re-cifrado, permutado aleatoriamente, y las salidas esta nueva secuencia de textos cifrados. Para probar que esto no altera el conteo de votos, cada servidor mix debe probar que cada boleta en las llegadas aparece en las salidas.

Más formalmente, cada servidor mix debe probar el siguiente enunciado NP:

$\exists\, \pi$ en $1, \ldots, n$nodos,
$\exists$ elementos $s_1, \ldots, s_n$,
$\forall\, i \in 1, \ldots, n$

$$C_{I,0} = (\alpha_{i,0}, \beta_{i,0}) \text{y} C_{\pi(i),1} = (\alpha_{\pi(i),1}, \beta_{\pi(i),1}),$$
Donde $\alpha_{\pi(i),1} = (\alpha_{i,0})(g^{s_i}), \beta_{\pi(i),1} = (\beta_{i,0})(y^{s_i})$.

El protocolo Chaum-Pedersen

Teniendo las dos fases en el sistema que son la de descifrado y r-ecifrado, para entender más la fase de re-cifrado se tiene un protocolo el cual contempla el re-cifrado manejando propiedades para este por lo que se maneja el protocolo Chaum-Pedersen para conocer algunas formas de usar el re-cifrado.

Un texto cifrado es re-cifrado de otro. Sea $c_1 = (\alpha_1, \beta_1) = (g^t, (m_1)(y^t)$ y $c_2 = (\alpha_2, \beta_2) = (g^u, (m_2)(y^u)$ cualesquiera dos textos cifrados. c_2 es un re-cifrado de c_1 si y solo si c_1 y c_2 son cifrados del mismo mensaje. Sea

$$(a_1, a_2, b_1, b_2) = \left(g, y, \frac{\alpha_2}{\alpha_1}, \frac{\beta_2}{\beta_1}\right) = \left(g, y, g^{u-t}, \frac{m_2}{m_1} y^{u-t}\right).$$

Afirmación

El texto cifrado c_2 es un re-cifrado de c_1 si y solo si $\log_{a_1}(b_1) = \log_{a_2}(b_2)$.

Prueba

Primero, se considera la implicación. Si c_2 es un re-cifrado de c_1, entonces $(a_1, a_2, b_1, b_2) = (g, y, g^{u-t}, \left(\frac{m_2}{m_1}\right)(y^{u-t})) = (g, y, g^{u-t}, y^{u-t})$ para $m_1 = m_2$. Así, se ve que $\log_g(g^{u-t}) = \log_y(y^{u-t}) = u - t$.

Se considera la implicación anterior. Si $\log_{a_1}(b_1) = \log_{a_2}(b_2)$, entonces $c_2 = ((\alpha_1)(g^v), (\beta_1)(y^v))$ para algún $v \in \mathbb{Z}_q$, pero este termino se puede factorizar fuera del texto cifrado para revelar $c_2 = (c_1)(g^v, y^v)$, donde (g^v, y^v) es simplemente un cifrado de 1!.

Así, probando que c_2 es un re-cifrado de c_1 inmediatamente se probara que $\log_{a_1}(b_1) = \log_{a_2}(b_2)$ en otras palabras que (a_1, a_2, b_1, b_2) es una cuarteta DDH. Se puede hacer esto corriendo el protocolo de Chaum-Pederson para probar que una cuarteta $(g, y, w, u) = (g, g^x, g^r, g^{(r)(x)})$ es una cuarteta DDH. Describiendo el protocolo

$$P \qquad\qquad V$$

$$s \in Z_q \xrightarrow[(a,b)=(g^s, y^s)]{} $$

$$\xleftarrow{c} \quad c \in Z_q$$

$$\xrightarrow[t=s+cr]{} \text{acepta si y solo si}$$

$$g^t = (a)(w^c) \wedge y^t = (b)(u^c)$$

Aquí, todo lo que el probador debe conocer es el factor r re-aleatorio; no el mensaje en si.

Considere un Mix nxn

Otra forma de utilizar el protocolo de Chaum-Pedersen es ocuparlo para servidores de n elementos utilizando el re-cifrado en ellos.

Con el protocolo de Chaum-Pedersen se construirá un mix nxn. Esto es, se probara que $(c'_1, \ldots, c'_n)$ es el re-cifrado de $(c_1, \ldots, c_n)$ sin revelar la correspondencia. Una guía es tener a los k servidores-mix probando el siguiente enunciado NP de conocimiento-cero:

$$\forall\, i \exists\, j \text{tal que } C_{i,k} \approx\ C_{j,(k+1)} \text{ y}$$

$$\forall j \exists\ i \text{ tal que } C_{i,k} \approx C_{j,(k+1)}$$

Desafortunadamente, servidores-mix nxn pueden probar que lo anterior sean ciertos y aún así fallar. Se considera el caso para $n = 3$. Suponga en las llegadas del servidor mix, hay dos votos para Dolores y uno solo para Martín. La salida de un servidor mix corrupto puede ser un voto para Dolores y dos para Martín y pueden seguir siendo verdad. Sin embargo una inspección rápida de las posibilidades muestra que la prueba es suficiente para un mix $2x2$.

3.1.7. Mix verificable Par a Par

Esta sección describe una forma de verificar todas las mix existentes para seguir teniendo una votación limpia

Para verificar las mix se deben revisar todas ellas por eso para mejor comprensión y facilidad de elaboración se dara primero la forma de verificar mix en forma par, después se generalizará para un mejor entendimiento del proceso

Se considera el caso verificable Par a Par con dos textos cifrados llegados, C_1 y C_2 y dos textos cifrados de salida C'_1 y C'_2. En este caso, la mix debe probar que:

$$(C_1 \approx C'_1) \wedge (C_2 \approx C'_2) \vee (C_1 \approx C'_1) \wedge (C_2 \approx C'_1)$$

Esto es equivalente a probar:

$$[(C_1 \approx C'_1) \vee (C_1 \approx C'_2)] \wedge [(C_2 \approx C'_1) \vee (C_2 \approx C'_2)] \wedge$$

$$\wedge [(C'_1 \approx C_1) \vee (C'_1 \approx C_2)] \wedge [(C'_2 \approx C_1) \vee (C'_2 \approx C_2)]$$

El protocolo honesto de conocimiento-cero Chaum-Pederson para probar que dos textos cifrados ElGamal, $C_1 = (\alpha_1, \beta_1) = (g^t, (m_1)(y^t)$y $C'_1 = (\alpha'_1, \beta'_1) = (g^u, (m_1)'(y^u)$ tienen el mismo texto llano (donde el que probara esto conoce el factor de re-cifrado, $v = u - t$). Sea $(a_1, a_2, b_1, b_2\ )$ el cuádruple $(g, y, \alpha'_1/\alpha_1, \beta'_1/\beta_1) = (g, y, g^v, (m'_1/m_1)(y^v)$. Entonces $m_1 = m_2$ si y solo si $\log_{a_1}(b_1) = \log_{a_2}(b_2) = v$. Para probar la igualdad, se usa el protocolo:

P		V
s selecciona s aleatoriamente de Z_q^* :	$\bar{A} = (\bar{A}_1, \bar{A}_2) = (a_1^s, a_2^s) \rightarrow$	
	$\leftarrow c$	: selecciona c aleatoriamente de Z_q^*
$r = s + c * v$	$r \rightarrow$	
		Acepta si $a_1^r = (A_1)(b_1^c)$ y $a_2^r = (A_2)(b_2^c)$

Ahora con el protocolo honesto de conocimiento-cero se probara la disyunción $[(C_1 \approx C_1') \vee (C_1 \approx C_2')]$ (donde el que lo probara conoce cualquiera el primer factor de re-cifrado r_1 o el segundo factor de re-cifrado r_2). Este protocolo es derivado del documento Cramer y amigos .

P		V
	$\bar{A}_1, \bar{A}_2 \rightarrow$	
	$\leftarrow c$	: selecciona c aleatoriamente
	$r_1, r_2, c_1, c_2 \rightarrow$	

El verificador acepta si ($C \approx D$) el verificador Chaum-Pedersen aceptaría la tripleta $(\bar{A}_1, c_1, r_1)$ con los textos cifrados C_1 y C_1' (g es un generador publico conocido de $\mathbb{Z}_q^*$) el verificador Chaum-Pedersen aceptaría la tripleta $(\bar{A}_2, c_2, r_2)$ con los textos cifrados C_1 y C_2' y ($\oplus$ la operación de adición en el anillo $\mathbb{Z}_q^*$) $c = c_1 \oplus c_2$.

Completez Sin perder la generalidad, el caso donde el que probara esto, conoce leyendo el factor de cifrar r_1. El que lo probara corre el simulador Chaum-Pedersen para los textos cifrados C_1 y C_2' para obtener la tripleta $(\bar{A}_2, c_2, r_2)$. El que lo probara escoge $\bar{A}_2$ como el probador honesto estaría en el protocolo Chaum-Pedersen y envía $\bar{A}_1$, $\bar{A}_2$ al verificador. Una vez que se recibió, prueba c con el verificador, el que va probar escoge c_1 tal que $c = c_1 \oplus c_2$ y calcula la respuesta r_1 para probar a c_1 como el probador honesto que estaría en el protocolo de Chaum-Pedersen. El verificador aceptaría porque $(\bar{A}_1, c_1, r_1)$ es construido honestamente como el protocolo de Chaum-Pedersen y $(\bar{A}_2, c_2, r_2)$ es construido con el simulador Chaum-Pedersen. Esto es, el verificador acepta $(\bar{A}_1, c_1, r_1)$ por que el protocolo Chaum-Pedersen esta completo y el verificador acepta $(\bar{A}_2, c_2, r_2)$ por que el protocolo Chaum-Pedersen es un honesto de conocimiento cero.

Indice de Sonido Recalcando que el sonido especial del protocolo Chaum-Pedersen implica que si para algún $\bar{A}$ se tiene más de un par valido r, c entonces se puede obtener un testigo para el hecho de que dos textos cifrados tienen el mismo texto llano (en particular, el factor de re-cifrado). Si en el protocolo el que va a probarlo, puede responder algunas ℓ fracciones de posibles cambios donde ℓ no es negligente, entonces se pone a prueba c aleatoriamente y en tiempo polinomial encontrar un par de pruebas $c \neq c'$ tal que el que va a probar pueda responder correctamente c y c'. Ya que $c = c_1 \oplus c_2$ y $c' = c'_1 \oplus c'_2$ y $c \neq c'$ entonces cualquiera de $c_1 \neq c'_1$ o $c_2 \neq c'_2$. Por lo tanto se puede extraer cualquier testigo, que $C_1 \approx C'_1$ o un testigo que $C_1 \approx C'_2$ y por lo tanto la disyunción $[(C_1 \approx C'_1) \vee (C_1 \approx C'_2)]$ será cierta.

Conocimiento-cero Honesto El simulador toma c_1 y c_2 independientemente aleatorios y selecciona c tal que $c = c_1 \oplus c_2$. El simulador construye $\bar{A}$ y r_2 correspondiente a la prueba c_2 del mismo modo que el simulador Chaum-Pedersen lo haría. Para c_1 y c_2 son escogidos independientemente, c es un elemento aleatorio de de $\mathbb{Z}_q^*$.

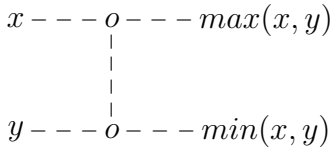
En el protocolo, se puede interpretar que c_1 y c_2 son partes del secreto c. Este es un caso especial de una conexión general entre los esquemas de partes secretas y la prueba de las formulas Booleanas monótonas. Ya que el protocolo de conocimiento-cero honesto, es también testigo indistinguible. Subrayando el testigo indistinguible prueba que puede ser compuesto en paralelo y recordar al testigo indistinguible.

3.1.8. Ir de Par a n

Esta sección describe lo que se busca la forma de verificar todos los votos entrantes en la redes-mix sin tener que hacerlo de par en par, ya que se vio como se realiza de par en par ahora se manejaran de tamaño mas grande.

Ya que se dio la forma par ahora solo se tiene que trasladar a la forma general, por eso gracias a la forma par vista ahora se entendera más fácil.

Se usaran conexiones de clases para construir mix verificable de n a n de muchos mix verificables Par a par. Una conexión de clases es un circuito con n cables de entrada y n cables de salida consistiendo de Par a Par compuertas comparadas. La operación de comparar es: Si los dos cables de entrada tienen valores x y y entonces el comparador muestra máx(x, y) en la cima de los cables de salidas y mín(x, y) en el fondo de los cables de salida.



Como una conexión de comparación es una conexión de clases valida si para cualquier conjunto posible de valores en n cables de salida, los valores en n cables de salida están ordenadas en clases. Existe conexión de clases con comparadores $(n)(\log^2 n$. Puede existir una conexión de clases más eficiente, pero son muy complejos. La propiedad de una conexión de clases es realizar todas las posibles permutaciones.

La estrategia es remplazar cada comparador en la conexión de clases con un mix verificable Par a Par para asegurar un mix verificable n a n. Esto es, (A) si, el que va probar, demuestra que cada comparador en la conexión esta haciendo una permutación correcta entonces la conexión entera debe hacer una permutación valida y (B) Ya que los comparadores en la conexión son capaces de realizar cualquier permutación n a n, el hecho de que la conexión de clases es usada para no dar ninguna información al verificador acerca de la permutación implementada.

- Ya que la prueba de la permutación es valida se sigue la estructura de la conexión de clases, esto es razonable para el que va ser la prueba de tomar en cuenta esto cuando se este haciendo la mix. Para hacer la mix seria de forma: Primero el que va hacer la prueba toma un reto-corto aleatoria para cada texto cifrado de llegada. Después el que va hacer la prueba corre la conexión de clases a una clase de acuerdo con el reto-corto. Después al final, el que va hacer la prueba borra los retos-cortos.

- Para obtener los retos para las pruebas, se puede tener cualquier grupo para retar contra reloj (es decir, tener cualquier grupo que sea parte del reto maestro) y después tener cada grupo posterior del primer paso de la prueba. Alternativamente, se puede usar el paradigma Fiat-Shamir y obtener los retos aplicando el primer mensaje del que lo va a probar en el protocolo. Adicionalmente, cuando se corra la red-mix, se puede pensar en tener cada servidor mix dando una prueba de eficacia antes del que el siguiente servidor mix comience a trabajar, pero en la práctica es mas probable para todos los servidores dar una prueba después de que toda la mezcla se ha hecho.

3.1.9. Remarcando tamaño en series

Esta sección hace mención del número de votantes que puede tener, o de la capacidad que puede tener la red-mix para mezclar.

Se tiene que tomar en cuenta en lo que respecta al tamaño de la mix ya que dependiendo del volumen del sistema que se tomara un tiempo determinado para poder verificar que la mix no se encuentre alterada.

En una gran elección probablemente no se pueda correr una sola red-mix que tenga a todos los votantes de la elección. Sin embargo, es probablemente mas razonable mezclar las boletas en series (posiblemente correspondiendo a alguna región geográfica).

El problema con hacer series muy largas, es el incremento en el tiempo para hacer la mix. El problema con hacer series mas pequeños es que los votantes son anónimos solo en sus respectivas series. Para mejor entendimiento de la eficiencia del esquema, con miles de votantes.

$$\begin{array}{rcl} \text{Número de votantes } (n) & = & 10^3 \\ \text{Número de comparadores } (n)(\log^2)(n) & = & 10^5 \\ \text{Modulo exponencial por comparador} & = & 10 \\ \text{Total Modulo Exponencial} & = & 10^6 \\ \text{Modulo exponencial por segundo (en PC)} & = & 50 - 100 \\ \text{Tiempo total para hacer mix verificable} & = & 3 - 4 \text{ horas} \end{array}$$

Se ve una falla razonable tener las computadoras separadas para cada recinto y $3 - 4$ horas es tiempo muy razonable para esperar los resultados de la elección. Por eso, se puede imaginar como actuarían los esquema actual en la práctica. Sin embargo, Andrew NET en su documento "Mix verificable del par ElGamal ", prueba un protocolo mix verificable mas rápido que requiere solo $8n+5$ modulo exponencial para probar y $9n + 2$ módulo exponencial para verificarlo. Dan Boneh dice que el protocolo mix de NET es suficientemente eficiente en una serie mix de 100000 boletas en cerca de 20 horas.

3.1.10. La red-mix de Neff

Esta sección menciona la forma de ir comprobando que lo que se va guardando este correcto, en la parte del texto cifrado.

La cual se utiliza para que el servidor pruebe que esten correctos los textos cifrados que se tienen.

Cada servidor-mix es también servidor descifrador (en una composición de cebolla). Durante la mezcla, el texto cifrado ElGamal $(g^r, (m)(y^r))$ re-aleatorian cada servidor escogiendo un c aleatorio y secreto , y produciendo $(g^{(c)(r)}$, $(m^c)(y^{(r)(c)}))$.

Adicionalmente, el par (g, g^c) es publicada por el servidor. Durante el descifrado, el servidor toma c raíces de cada texto cifrado y prueba que sean correctos demostrando igualdad del logaritmo discreto (por ejemplo, vía Chaum-Pedersen) usando el par publicado (g, g^c).

3.1.11. Verificación en serie

Esta sección maneja la forma de verificar teniendo una gran cantidad de votos, es decir, mediante series se comrpobara la veracidad del voto entrante.

Ya que las votaciones no se dan a cuenta gotas, es decir que se van a manejar muchos votos se requiere verificar un número grande que son ellos para eso una forma de verificarlos mas rápido es en forma de serie.

El problema abstracto dice (fuera del contexto de votación): dando g y n pares (x_i, y_i), verificar que cada $y_i = g^{x_i})$. La solución clave es hacer n exponenciales y comparar el resultado con las $(y_i)(s)$. Pero considerando el caso $n = 2$: se puede salvar una exponencial revisando si $(x_1 + x_2, (y_1)(y_2))$ es un par buen. Ciertamente si los dos pares originales son buenos, el tercero lo será.

La conversión no es en general, pero con aleatoriedad se puede tener una oportunidad de encontrar un par malo. El trabajo en general es tomar un subconjunto aleatorio $S \subseteq [n]$, combinar, y revisar. Es decir, revisar que

$$\left(\sum_{i \in S} x_i, \prod_{i \in S} y_i\right)$$

Es un par bueno, usando solo una exponencial.

Teorema 3.1. *Si el conjunto original de pares no es valida, entonces revisar la detección de este hecho con probabilidad a lo más* $1/2$.

Es fácil ver que la probabilidad de detectarlo no puede ser mejor que $1/2$. Si solo hay un par malo en el conjunto, será excluido la mitad del tiempo, en cuyo caso pasa la prueba. La prueba en general es: se escoge un bit aleatorio simple de un par i (es

decir, incluido i en S, o no), toma un s_i aleatorio de algún rango pequeño $0, \ldots, 2^{t-1}$. Después se revisa que

$$\left(\sum_{i \in [n]} (s_i)(x_i), \prod_{i \in [n]} y_i^{s_i} \right)$$

es un par válido.

En términos de cálculo, esta prueba solo toma $1{,}5t$ multiplicaciones (para calcular $y_i^{s_i}$) por par, pero solo una exponencial en general. Esta prueba tiene algunas salidas en la aplicación a las redes-mix: para el primera prueba, la mix solo necesita revelar el conjunto de salida de textos cifrados correspondiente al conjunto aleatorio S.

Esto compromete un poco la seguridad del votante, pero no demasiado. En la segunda prueba, la mix debe revelar el conjunto de salidas de textos cifrados correspondiente a cada uno de los 2^t valores posibles de s_i. Este enunciado significa más información acerca de cómo los votantes permutan con la red-mix.

Teorema 3.2. *La prueba solo acepta llegadas invalidas con probabilidad a lo más* 2^{-t}

Prueba

Si el conjunto es invalido, sea un solo par invalido $(\bar{x}, \bar{y})$ y sea $\bar{s}$ el correspondiente coeficiente aleatorio. Entonces la prueba revisa si $(\bar{s}\bar{x} + \sum s_j x_j, \bar{y}^{\bar{s}} \prod y_j^{s_j})$ es un par valido.

Sea $\bar{z} \neq \bar{x}$ el logaritmo discreto de $\bar{y}$, base g. Entonces el logaritmo discreto de la parte derecha del par es $\bar{s})(\bar{z}) + c$, donde c es alguna constante independiente de $\bar{s}$.

Para pasar la prueba, se necesita $(\bar{s})(\bar{x}) + \sum (s_j)(x_j) = (\bar{s})(\bar{z}) + c$, donde la suma es independiente de $\bar{s}$. Porque g es de primer orden y $\bar{x} \neq \bar{z}$, hay un solo valor de $\bar{s}$ que permite pasar la prueba, y es escogido con probabilidad 2^{-t}.

3.1.12. El Boneh-Golle aproximado a las redes-mix

Esta sección describe la forma de obtener una verificación de el número de votantes que entre debe salir, esto nos da ventaja ya que mediante esta prueba se puede dar cuenta de alteraciones

Otra forma de revisar la honestidad o la veracidad de la seguridad en los elementos de la red es mediante la prueba de Bohen-Golle, revisando la correspondencia de votos emitidos con los votos publicados.

Se quiere una mejoría al verificar que una mix esta operando adecuadamente, es decir, que permuta sus llegadas y las re-cifrarlo usando una hoja de suma aleatoria en el exponente. La idea es: tomar un subconjunto aleatorio S de las llegadas, y tener los textos cifrados c_i de las llegadas para cada $i \in S$. Se combinan todas las llegadas, vía componentes multiplicativos, para tener $\bar{c}$, una meta inicial que es un texto cifrado de $\hat{m} = \prod_{i \in S} m_i$.

Probar al servidor mix para producir el conjunto correspondiente S, del mismo tamaño de S tal que la meta final (definida similarmente) cifra el mismo mensaje $\hat{m}$. (el conjunto S podría ser permutado así como no revelar la correspondencia exacta entre llegadas y salidas de los textos cifrados). La prueba, como siempre, es demostrar la igualdad de los logaritmos discretos.

Hay una gran diferencia, de lo que es verdad acerca de las metas de salida, y que es probable por el servidor mix. Por ejemplo, es posible que la mix cambie algunos votos (cambie 10 votos para Carlos y 10 votos para Enrique en 11 votos para Carlos y 9 votos para Enrique), casi siempre se puede encontrar textos cifrados de salida tal que la meta de salida cifre el mismo mensaje de meta como la meta de salida. (En el ejemplo, el único problema del conjunto S es el correspondiente a los 10 votos para Enrique).

El teorema informal es que siendo difícil probar la mezcla no valida sea correcta. Formalmente:

Teorema 3.3. *Si el servidor mix opera incorrectamente, y si la mix puede satisfacer una prueba aleatoria con probabilidad a lo más* $3/8 - \sigma$*, entonces el logaritmo discreto puede resolverse en tiempo polinomial.*

Se pierde parte de la seguridad del votante con cada repetición de la prueba: con α pruebas, se puede acercar a la correspondiente llegada del texto cifrado para abarcar $n/2^{\alpha}$ de las salidas. Quizás la serie natural de mix pueda esconder mejor la correspondencia global de llegadas y salidas, pero esto es difícil de analizar.

Para arreglarlo: aplica la idea detrás de la mix 2 x 2. Se escoge un conjunto S y se combinan los textos cifrados en dos metas de llegada (correspondientes a S y $\bar{S}$) y dos metas de salida (correspondientes a T y $\bar{T}$), y demostrar que los cuatro

conjuntos tienen el mismo tamaño y que la meta de salida cifra los mismos valores que en la meta de llegada (pero sin dar la correspondencia).

¿ El esquema sigue siendo seguro? La prueba del teorema 3 depende del uso de un conjunto aleatorio S, pero la técnica solo funciona si $[S]$ es exactamente $n/2$ (de otra forma es claro como la meta de salida corresponde a la meta de llegada).

3.1.13. Revisión parcial aleatorio (RPC)

Esta sección hace enfasis a una de las vetajas con mayor peso en la votación que es la verificación parcial del sistema, ya que mediante esta forma de revisión se puede examinar como se realiza la votación cuando uno quiera.

Una de las ventajas que se tiene el sistema de votación es la revisión constante de todo el sistema en el momento que uno quiera, una de las revisión que se puede realizar frecuentemente es a los servidores mix.

Cada organización que hace la mezcla actualmente controla dos mix consecutivos, y dos mix normales para ambas. Las pruebas son salidas a la mitad; esto es, un subconjunto aleatorio S se escoge de en medio de los textos cifrados. Para cada texto cifrado en S, su correspondiente texto cifrado de entrada es identificado y revisado. Para cada texto cifrado en $\bar{S}$, le corresponde un texto cifrado de salida que es identificado y revisado.

Hay algunas perdidas de seguridad: al cruzar dos mix consecutivamente, se puede aprender que el texto cifrado de llegada corresponde a uno de los $n/2$ textos cifrados de salida. Una propiedad de sonoridad: cuando corrompen, la oportunidad de ser atrapados es exponencial (base $1/2$) al número de votos que la mix intenta cambiar.

Esto es razonable para grandes elecciones, hace pensar que no va decidir para algunos votos solamente.(En este caso, se puede anexar una de las técnicas de prueba lentas en los datos originales). Para la eficiencia, se pueden hacer verificación en serie de los correspondientes textos cifrados. Esto no causa que se pierda la privacidad, pero si afecta la sonoridad.

3.1.14. Verificación del votante en el sistema de votación red-mix

Esta sección describe la forma de verificar la forma de votar del usuario, claro mediante caracteristicas de uso que se describen en esta misma sección: las partes

que la componen es: de notación, construcción y el protocolo.

Se tiene ahora las fases para el sistema, ahora se centrara la explicación en la fase de cifrado para lo cual, primero se plantean algunas notaciones las cuales se ocupara, después se explicarán las dos herramientas más importantes que se ocuparan en la contrucción. Seguido del planteamiento del protocolo que se necesita para realizar la fase de cifrado.

La fase de cifrado: esta fase, es opuesta a la fase mix, no puede ser verificada públicamente, por que si se verifica públicamente puede ser usado como un recibo y votar. Así, esta fase puede verificarse en la votación.

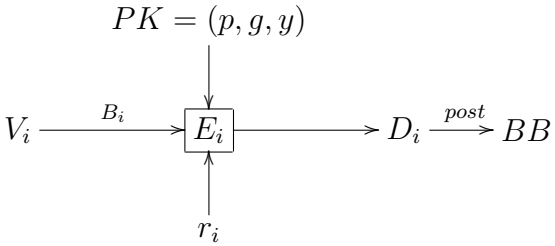
El votante es humano, y así la fase de cifrado requiere ser verificado por un humano. Construir cualquier protocolo que un humano pueda verificar es muy engañoso, lo opuesto a los servidores, son muy limitados calculando.

Chaum propone una fase de cifrado que es verificado en la votación sin verificarlo publicamente. Para esto se requieren dos herramientas: criptografía visual y técnicas de cortar y escoger. Se presenta una variante del esquema de Chaum.

Notación

- V_i denota los i votantes.
- B_i denota las boletas de V_i.
- E denota la caja de cifrado en la estación de votación, que cifra aplicando el esquema de cifrado ElGamal.
- D_i denota el cifrado de B_i. Esto es, $D_i = (g^{r_i}, (B_i)(y^{r_i}))$.

Para cada i, (V_i, D_i) es anunciada en un pizarrón publico. Así, el proceso de cifrado es de la forma:



La meta es construir una fase de cifrado que cumpla las siguientes propiedades.

- El votante tiene la confiabilidad que D_i representa a B_i, es decir, que E actúa fielmente en su nombre.
- El votante no entrega un recibo que permita probar a otros por cual voto.
- Si E se comporta sospechoso entonces el votante tiene una prueba irrefutable de su comportamiento (y el visitante no puede producir una prueba si E no se comporta sospechoso).

Construcción

Ya que se requiere dos herramientas esenciales para la contrucciones que son la tecnica de cortar y escoger, al igual que la de criptografía visual, que se describen para su mejor comprensión detenidamente.

Esta construcción es una variante de la construcción de Chaum. Dos herramientas se usan:

La técnica de cortar y escoger E genera un recibo que puede separar en dos partes. El votante puede observar los dos pedazos pero solo puede guardar uno, el que él escoga. Esta idea se maneja como: E, en las llegadas una boleta B_i, primero se cifra B_i para obtener un texto cifrado $C_i = (g^{r_i}, (B_i)(y^{r_i}))$, y después se re-cifra C_i para obtener un texto cifrado $D_i = (g^{r_i+s_i}, (B_i)(y^{r_i+s_i}))$. E crea un recibo que consiste de dos partes; la primera parte es la tripleta (B_i, C_i, r_i) y la segunda parte es una tripleta (C_i, D_i, s_i). El votante guarda la de su elección. La idea es que si E se comporta sospechoso enmtonces, con la probabilidad de $1/2$, el votante tiene una prueba del mal comportamiento.

Hay algunos problemas con la aproximación.

1. La tripleta (B_i, C_i, r_i) es esencialmente un recibo que el votante V_i usa la boleta B_i.
2. Para verificar, con probabilidad $1/2$, que su voto fue contado correctamente, el votante necesita revisar que C_i en la tripleta (B_i, C_i, r_i) es la misma que C_i en la tripleta (C_i, D_i, s_i), no esta claro como un humano puede probar que los dos textos cifrados son iguales.
3. Cualquier votante corrupto puede crear una tripleta de su elección, y después convencer que la tripleta se la dio E.

Hay muchas maneras de abordar estos problemas. En particular, el primer problema se puede resolver teniendo un bucle lleno de las tripletas de la forma (B_i, C_i, r_i) en la estación de votación, y así cualquier votante puede tomar

cualquier tripleta, observando su voto.

Otra solución es programar a E que de al votante la tripleta (B_i, C_i, r_i) para cualquier B_i de su elección, observando su voto. El segundo problema, se puede resolver usando criptografía visual, y el tercer problema se puede resolver teniendo un E que firme todas las tripletas que genere. (Criptografía visual permite que los recibos se vean mas aleatorios).

Criptografía Visual Esta herramienta es usada para ayudar al votante resolver computacionalmente cosas que se ven difícil para el. Mas específico, usando la criptografía visual, un votante puede calcular la función xor. Esto se hace transformando cada cadena de bits en una transparencia, donde cada bit es representada por una matriz de $2x2$:

Entonces, si $B_i = B'_i \oplus B''_i$, después cuando se alinien las dos transparencias, correspondientes de B'_i y B''_i una encima de la otra, obtenemos B_i. Así los votantes pueden usar su sentido visual para verificar que $B_i = B'_i \oplus B''_i$.

El protocolo

Ahora se explicará el algoritmo a utilizar para formar la fase de cifrado, al igual que su verificación por parte del votante.

El protocolo E usa el algoritmo de cifrado ElGamal y dos algoritmos de firma; un algoritmo inicial de firma σ_I, y un algoritmo final de firma σ_F.

1. Votante V_i entrega su boleta B_i.
2. E escoge un r_i y s_i aleatorios, y calcula $C_i = (g^{r_i}, (B_i)(y^{r_i}))$ y $D_i = (g^{r_i+s_i}, (B_i)(y^{r_i+s_i}))$. Sea:
   - $R_i = (V_i, D_i)$
   - $S_i = (B_i, C_i, r_i)$
   - $T_i = (C_i, D_i, s_i)$
   - El votante tiene R_i, S_i, T_i todas firmadas con σ_I (la firma es para asegurar que R_i, S_i, T_i fueron realmente dadas por la maquina).
   - El votante revisa que $B_i(S_i)$ sea su proyección del voto. Si no es la proyecció n del voto, entonces el votante simplemente vuelve hacer el proceso de votación.
   - El votante deberia también revisar que se cumplan las dos igualdades:

- $C_i(S_i) = C_i(T_i)$
- $D_i(T_i) = D_i(R_i)$

  Si alguna de estas igualdades no se cumple entonces el votante se queja. De otro modo, el votante dice ‘ ‘abandono‘ ‘. Para un humano seria dificil checarlo. Por eso se implementaria la criptografía visual; así también los recibos serian más aleatorios.
- $\sigma_F(R_i)$ es dado al votante, y $(R_i, \sigma_F(R_i))$ es anunciada en el pizarrón. $(R_i, \sigma_F(R_i))$ es la boleta oficial.
- El votante escoge el que desea guardar (R_i, S_i) o (R_i, T_i) (el no puede guardar ambos).
- El votante abandona el complejo, y revisa que todas las firmas que recibio son validas (con respecto de σ_I y σ_F).
- El votante puede revisar que su boleta R_i aparece en el pizarrón.

Corrección Si $B_i \rightarrow C_i \rightarrow D_i$ no es un correcto cifrado y re-cifrado ElGamal, será detectado con probabilidad a lo mas 1/2.

El votante no tiene recibo El $S_i s$ (aun firmado por σ_I) se puede obtener fácil. Esto se puede hacer teniendo un bucle de todas los $S_i s$, o programando E para dar la tripleta del voto (B_i, C_i, r_i), para cualquier B_i de su elección, observando su voto, Así, S_i no es necesario relacionarlo con el voto del votante. Dando T_i no hay forma de verifcarlo (sin conocer la clave secreta ElGamal) que B_i es el texto llano correspondiente a T_i. Así ni S_i o T_i pueden ser usados como recibos.

3.1.15. Conclusión de la variante Ron en el esquema de Chaum

Esta sección describe la forma de manejar la elección mediante uso de electronica para ello se dara a conocer un esquema el cual facilitara la forma del usuario a comprobar, al igual que al sistema de corroborar lo que el usuario disponga, mediante el esquema de Chaum.

El objetivo que se quiere alcanzar con estos esquemas es que la elección pueda ser manejado electronicamente, al igual que verificable en cada paso que se dé de la votación. Una vez conocidas las herramientas a utilizar, se explicará como se realizaría la votación mediante el uso de las herramientas, puntualizando más en el uso de la criptografía visual. Después como el protocolo que se manejo es una variante del esquema de Chaum entonces se enunciará esa pequeña diferencia con el esquema.

El propósito detrás de este esquema es tener una elección que pueda ser manejado electrónicamente, pero puede ser verificado en cada paso.

Un voto para un votante llamado V_i en la variante es construida vía un engaño $B_i \rightarrow C_i \rightarrow D_i$, donde B_i es la boleta original, C_i es el primer cifrado ElGamal $(g^{r_i}, (B_i)(y^{r_i})$, y D_i es el re-cifrado $(g^{r_i+s_i}, (B_i)(y^{r_i+s_i}))$.

El voto completo esta formado de B_i a D_i, pero se dejara al usuario tomar $B_i \rightarrow C_i$ o $C_i \rightarrow D_i$. Este toma uno de ellos al azar, y también la maquina tiene $1/2$ de posibilidad de ser atrapada si estuviera fallando.

El recibo es $R_i = (V_i, D_i)$ el cual consiste de su nombre y su boleta cifrada, más la firma en R_i; $\sigma_I(R_i)$, $\sigma_F(R_i)$, donde σ_I es generada por la maquina para cada R_i de salidas (y deben ser más de una por si el votante cambiara de idea antes de emitir su voto), y σ_F es generada después de que el usuario tome su decisión de emitir su voto (esto es mas frecuente cuando o donde ocurra un engaño). La evidencia presentada al usuario es:

$$S_i = (B_i, C_i, r_i), \sigma_I(S_i)$$
$$T_i = (C_i, D_i, s_i), \sigma_I(T_i)$$

El usuario emite su voto, y recibir $\sigma_F(R_i)$, que también es anunciada en algún pizarrón. El usuario tiene que tomar con su R_i, $\sigma_I(R_i)$, $\sigma_F(R_i)$, y cualquiera $(S_i, \sigma_I(S_i))$ o $(T_i, \sigma_I(T_i))$. Los guardianes en la estación de votación se aseguran que la otra se destruya. En casa (o vía algún servicio público) el usuario deberá verificar que la parte que el tiene es consistente. Si no fuera así, tendría que quejarse y presentar la evidencia que tiene.

Si la máquina engaño significativamente, suficiente para llamarlo error se tendrá que hacer otra votación. Si solo una queja es registrada, aun así todas las firmas son validas, es más factible que el usuario este engañando. El esquema simplificado para introducirse el esquema de Chaum:

- El usuario puede probar funcionalmente el corte y escoja (actuando como impredecible origen de aleatoriedad).
- Se puede permitir a los votantes generar muchos votos, pero solo el último se cuenta (en caso de que tengan que presenten recibos falsos a terceras personas).

  Hay un debate, si permitir a los usuarios escoger el recibo que se llevaran. Un recibo claramente contiene su boleta, en otros lados otros observarían alea-

toriamente en la basura. En este escenario, se puede dar el caso que muchos usuarios escojan la parte que no esta en la basura. Para resolverlo, se introduce el truco final: usando criptografía visual.

Se asume que la máquina no fue alterada con los votantes. En este caso de generar el engaño $B_i \rightarrow C_i \rightarrow D_i$, se generan dos engaños con *xor* a los valores originales. Uno de los engaños es justo un valor elegido al azar:

$$
\begin{array}{c}
B'_i \xrightarrow{r'_i} C'_i \xrightarrow{s'_i} D'_i \\
\oplus \\
B''_i \xrightarrow{r''_i} C''_i \xrightarrow{s''_i} D''_i \\
= \\
B_i
\end{array}
$$

Lo imprimen en dos transparencias entonces cuando se pone una encima de la otra, el usuario puede verificar que B_i esta formado como el *xor* de B'_i y B''_i que observa que corresponda con su elección. Entonces se lleva a casa uno de las dos transparencias conteniendo cualquiera (B'_i, r'_i, s''_i) o (B''_i, r''_i, s'_i),tan bien como los valores originales $R_i = (V_i, D'_i, D''_i)$ y $\hat{c}_i = (C'_i, C''_i)$.

Los D_i son guardados juntos en la red-mix, pero en la fase de revelación, solo se revela B_i, no las dos partes (de otro modo podrían generar recibos). Tristemente, probar el descifrado correctamente en el esquema se ve difícil.

Esquema de Chaum

Una vez visto la variante que se ocupará del esquema original, se describirá el esquema de Chaum del cual fue sacado la variante que se manejo para la fase de cifrado.

El esquema original de Chaum es similar a la variante, pero muy difícil de seguir.

Cada parte tiene un valor q que es un número en serie asociado con su voto. Generan la imagen de la boleta $B \in \{0,1\}^{m \times m}$. Para establecer el voto, la maquina genera dos transparencias, pero se quiere asegurar que las dos transparencias tienen la información suficiente sobre la boleta tal que la partes correctas puedan salir del voto solo de una transparencia. Se tendrán entonces generadas una parte blanca y

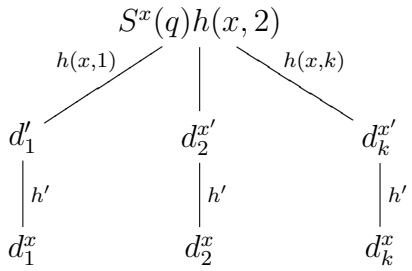
una parte roja, y se vera la transparencia a la mitad de arriba o la mitad de abajo.

La parte blanca (W) se ve como un dato aleatorio. Después se calcula la roja $R = W \oplus B$. Ahora se genera la parte de arriba y abajo y aseguran que ambas contengan información sobre B. Sea $B = (B^t, B^b)$ donde cada una tiene la suficiente redundancia tal que B pueda ser construida enteramente por cualquiera. Los valores L_t y L_b son las dos capas (es decir, lo que se imprime en la transparencia de arriba y abajo).

$$R^t = W^b \oplus B^t$$
$$R^b = W^t \oplus B^b$$

$$L^t = R^t \oplus W^t$$
$$L^b = W^b \oplus R^b$$

Cuando las dos capas están en la cima de la otra, la imagen de la boleta es visible. Si el votante aprueba la boleta, seleccionara la capa x al azar. La máquina le añade q, $s^x(q)$ (donde $s^x()$ es la firma y también es usada para generar W^x). Finalmente, se añadira una cebolla pre-mix (D_k^t, D_k^b). Una transparencia es destruida, y la otra es publicada. Las W son generadas en sentido que permiten usar una red-mix descifrado para revelar las boletas.



Después el conjunto $W^x = d_1^x \oplus d_2^x \oplus d_k^x$.

Cada $d_i^{x'}$ es usado en uno de los mix descifrados. Se refiere a los valores intermedios como muñecas , y generarlos de la forma:

$$\begin{aligned} D_0 &= 0 \\ D_l &= (e_l)(d_l^{x'} \parallel D_{l-1}) \\ B_k^x &= R^x \\ B_l^x &= B_{l+1}^x \oplus d_l^x \\ B_0^x &= B^x \end{aligned}$$

Cada máquina red-mix en turno recibe (D_l, B_l) y puede remover una capa de la cebolla revelando el $d_l^{x'}$ local mediante su clave privada. Esto puede entonces resumir el valor para tener d_l^x el cual puede ser usado para recuperar B^x. Esto puede sacar (D_{l-1}, B_{l-1}). Como siempre, se deberá adicionar probablemente el resultado usando técnicas .

3.2. Cifrado homomorfo

En esta sección se dará a conocer una forma de conservar intactos los elementos sin que estos sean alterados al momento de estar trabajando con ellos para lo cual se usará el cifrado homomorfo. Se describirán lo que hacen los cifrados homomorfos y se explicará el protocolo Paillier.

Como se quiere la seguridad en los votantes, es decir, el anonimato en muchos casos lo que se plantea es que mediante el cifrado homomorfo se pueda asegurar esta caracteristica, ya que los elementos privados son ocupados para realizar operaciones con ellos para su verificación, lo cual hace que en algunos casos se descubra la identidad del elemento, para eso se ocupara el cifrado homomorfo para mantener la privacidad en los elementos al ser manipulados en las operaciones. Para lo cual primero se plantearan los objetivos del cifrado, para seguir con la forma del algoritmo, es decir las propiedades a cumplir, y ya que utiliza dos esquemas una de ellas firmas ciegas visto en la introducción, y el criptosistema Pallier el cual se describirá y planteará su sistema el cual también ocupa la propiedad de seguir manteniendo la privacidad de los elementos a utilizar. Ya que se conocen las herramientas a utilizar, se veran los resultados o como el cifrado es visto de tal maner a que mediante la publicación que se realice de los resultados se explicará la forma de como actua este cifrado en los votos hechos y como impedir su alteración para los resultados finales.

Lo que se quiere al final es tener un sistema de votación donde:

a) cada votante pone exactamente una boleta y

b) el voto es anónimo.

Para que se lleve acabo los puntos anteriores se utilizaran: Firmas Ciegas, que permiten una votación anónima, y el Criptosistema Paillier, que permite resumir votos a pesar de que han sido cifrados. A partir de ahí, se intenta sacar un sistema de votación sin seguridad y como se puede añadir la seguridad sin alterar el esquema simple.

3.2.1. Protocolos de votación con cifrado homomorfo

El cifrado homomorfo es utilizado para mantener los elementos utilizados dentro de una operación realizada con estos elementos, en otras palabras, el cifrado homomorfo permite hacer cualquier operación de los elementos sin que la operación altere o descubra a los elementos que se utilicen.

- Cifrado Homomorfico El algoritmo de cifrado $E()$ es homomorfico si, dando $E(x)$ y $E(y)$, se puede obtener $E(X \perp Y)$ sin descifrar x; y para alguna operación $\perp$.
- RSA (Homomorfico Multiplicativo) Dada$c_i = E(m_i) = m_i^e \mod N$

c_1	$=$	$m_1^e \mod N$				
c_2	$=$	$m_2^e \mod N$				
$c_1 * c_2$	$=$	$m_1^e * m_2^e \mod N$	$=$	$(m_1 * m_2)^e \mod N$		

Lo que da entender la tabla es una propiedad de homomorfismo multiplicativo donde: $E(m_1) * E(m_2) = E(m_1 * m_2)$, esto quiere decir, que teniendo dos elementos, al querer realizar una operación en este caso una multiplicación se conservaran los elementos originales.

3.2.2. Criptosistema Paillier

Una de las herramientas a utilizar es el critosistema Paillier la cual también va a trabajar mediante la conservación de la privacidad en los elementos que se les aplica la operación, para lo cual se planteará como cifrar y descifrar sabiendo esta propiedad muy importante.

Generación de claves

Como en RSA, se escogen dos primos p, q y se calcula $n = p * q \mod n^2$. Se nota que $\varphi(n^2) = n * \varphi(n) = n * \varphi(p) * \varphi(q)$ y todos los elementos tienen orden divisible a $\varphi(n^2)$. Crear $C.P. = (n, g)$ donde g de orden múltiplo a N y $C.P. = (\lambda(n))$, donde $\lambda(n) = mcm(p-1, q-1)$

Cifrado

Para cifrar un mensaje $m \in Z_n$:

- Se Elije $x \in Z_n^*$
- Producir el cifrado $E(m) = g^m x^n \mod n^2$.

Haciendo aritmética, se puede afirmar que Paillier tiene las siguientes propiedades útiles:

$$\begin{array}{lcll} E(m_1) & \cong & g^{m_1}x_1^n \mod n^2 & \\ E(m_2) & \cong & g^{m_2}x_2^n \mod n^2 & \\ \hline E(m_1) * E(m_2) & \cong & g^{m_1+m_2}(x_1 * x_2)^n \mod n^2 & = \quad E(m_1 + m_2) \end{array}$$

Se tiene dos elementos que si se les aplica la operación en este caso de la suma van a conservarse los elementos iniciales sin que la operación los altere, esta propiedad se tiene por el cifrado homomorfo

Descifrado

Utilice $L(u) = \frac{(u-1}{n}$ para $u \cong 1 \mod n$ La formula para $L(u)$ no es módulo algo. Si $c = E(m)$, entonces

$$m \cong \left(\frac{L(c^{\lambda(n)} \mod n^2)}{L(g^{\lambda(n)} \mod n^2} \right) \mod n$$

Beneficios de la Criptografía Paillier

- Permite la propiedad de homomorfismo que se quiere para votar
- Es semánticamente seguro, asumiendo que es dificil distinguir los n residuos de los n no residuos $\mod n^2$. Semánticamente seguro quiere decir que no se puede distinguir $E(0)$ de $E(1)$ mejor que 50 % cuando $n \to \infty$

3.2.3. Votación con Criptosistema Paillier y Firmas Ciegas

Ya que se tienen las dos bases para la contrucción se planteara la manera de utilizar estos sistemas donde se observa el resultado, al igual describir como verificar por parte del votante no interfiera por este lado del cifrado puesto en las publicaciones hechas.

Motivación: pizarrón de votación

Los resultados a los que se quiere llegar son vistos en la publicación de los votos ya que a cifrados de votantes deben obtenerse votos diferentes, para cada votante que lo ha hecho para eso utiliza, entre otras cosas para que los votantes no tengan un mismo cifrado todos.

Se usa $E(m_1) * E(m_2) = E(m_1 + m_2)$ para publicar los votos. Si se tuviera un pizarrón muy grande con los candidatos X, Y y Z y los votantes $V_1, \ldots, V_n$.

	X	Y	Z
V_1	1	0	0
V_2	0	1	0
V_3	1	0	0
Total	2	1	0

Con un pizarrón normal las boletas estan en texto llano. X, Y y Z son los candidatos. Los V_i son los votantes. La primera entrada representa un voto para el candidato X.

	X	Y	Z
V_1	C_{1X}	C_{1Y}	C_{1Z}
V_2	C_{2X}	C_{2Y}	C_{2Z}
V_3	C_{3X}	C_{3Y}	C_{3Z}
Total	C_X	C_Y	C_Z

C_{1X}, etc., son los votos cifrados usando un esquema de clave pública, con la clave privada del oficial de la elección.

Si se tiene la propiedad del homomorfismo, se necesita el decifrado de las figuras en la fila del total, esto probara la seguridad del voto. Esto es encontrar $E(T_j) = \prod_i E_{ij}$

El cifrado tiene que ser aleatorio. El cifrado de los 0 tiene que ser diferente, para que los adversarios no sean capaces de contar que cualquier de las dos personas hizo el mismo voto, tan solo revisando cualquiera de los textos cifrados sean iguales. Similarmente el cifrado de los 1 también se debe ver diferente. Por lo tanto un RSA simple no es apropiado para la aplicación.

Corrección

Ahora que estan las dos herramientas, Criptosistema Paillier y Firmas ciegas, regresando al pizarrón público de votos.

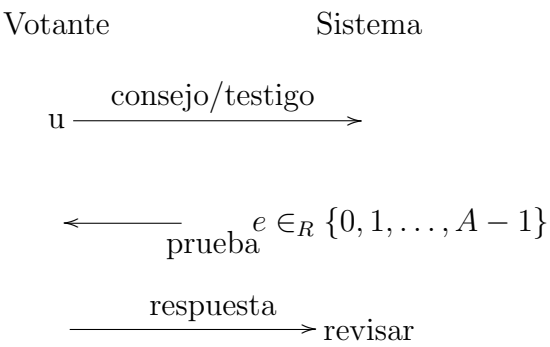
- Para evitar que un votante emita múltiples votos para un candidato (o votos negativos) se debe asegurar que $m_{ij} \in 0; 1$ y que la fila del subtotal $\in 0; 1$.

- Que se cumpla no maleabilidad. Por ejemplo, si el voto de Lupe fuera $m_{ij} \in 0; 1$, Ernesto podría negar la votación de Lupe calculando:

$$\frac{E(1)}{E(m_{ij}} = E(1) * E(-m_{ij}) = E(1 - m_{ij})$$

  La solución es usando la prueba de conocimiento-cero

- Para probar que el voto sea correcto:



  Este ciclo se hace t veces.

El truco Fiat-Shamir: remplazar $e \in_R 0, 1, \ldots, A-1$ con $e = \text{hash(consejo)}$ para hacer una prueba ZK no interactiva. Por lo tanto, *consejo/testigos* y la *respuesta* puede reducirse a un mensaje.

• Se demuestra como el votante puede probar que conoce por cual votó (conoce m), cuando se usa el cifrado Paillier.

$$c = g^m x^N \mod N^2$$

Votante		Sistema
$r \in_R Z_N$		
$S \in_R Z_N^*$	$\xrightarrow[u=g^r s^N \mod N^2]{}$	
	$\xleftarrow{e}$	$e \in_R 0, 1, \ldots, A-1$
$v = r - em$		
$w = sx^{-e} g^{(r-em) \div N}$	$\longrightarrow$	revisar $g^v \zeta_{e,v}^e w^N =^? u$

La revisión sería

$$g^{r-em}(g^m x^N)^e (sx^{-e} g^{(r-em)\div N})^N =^? u \longleftrightarrow g^r s^N =^? u$$

Después de t pruebas superadas, la oportunidad de falsificar es $\simeq \frac{1}{A^t}$

¿Comó se detiene el descifrado de una entrada individual con la clave secreta d? ¿Se puede controlar la clave secreta?. Usar partes secretas para dividir d tal que cualquier parte t es usada para volver hacer d. Desafortunadamente, esto todavía deja la generación de claves y descifrado (combinación de partes) con puntos vulnerables.

Conclusiones

La importacia de las elecciones es relevante en nuestro país, tanto en la vida política como en los grupos en donde se eleige a sus representantes de este modo. Una enorme ventaja de los protocolos de votación electrónica es que se puede efectuar a través de internet y la seguridad (anonimato, privacidad, no duplicidad de votos de un mismo votante, etc.) queda garantizada por los elementos criptográficos involucrados. Otra enorme ventaja radica en la velocidad de la obtención de resultados del escrutiño de los votos. Gracias al uso de las redes Mix no se pierde el anomimato de los votantes y si se garantiza que cada votante emita a lo más un único voto. Con los parámetros adecuados en los protocolos intrínsecos de la votación electrónica, se garantiza la inviolabilidad y veracidad (votos apócrifos son identificados e invalidados) de los votos recolectados. Una ventaja más es el enorme ahorro en papel y la disposición de las "boletas electrónicas" en internet para los votantes, lo que podría inhibir en cierto grado el abstencionismo.

Esto nos hace ver que la votación electrónica es una herramienta muy útil por la situación que se esta presentando en el país sobre todo por el gran abstencionismo que existe, una causante de esta situación es la ausencia de la gente en las casillas. Algunos dicen que por falta de tiempo no pueden votar, otros por lo lejano que se encuentran las casillas. Para estas situaciones y otras serian abordados por la votación electrónica. Aunque el principal problema es la falta de credibilidad en los candidatos.

Cabe destacar la importancia de implementaciones de protocolos de votación electrónica, implementación de una infraestructura de clave pública, así como la creación de políticas para un correcto uso y funcionamiento de votaciones electrónicas, aunque estos temas quedan fuera del alcance del presente trabajo.

Bibliografía

[BSW06] A. Beutelspacher, J. Schwenk, and K. Wolfenstetter. *Moderne Verfahren der Kryptographie, Von RSA zu Zero-Knowledge*. Mathematik. Vieweg, 2006.

[Cha85] De Chaum. Manufacturing security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28:1030–1044, 1985.

[Coh96] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics. Springer-Verlag, 1996.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[RS04] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal Roy and Willi Meier, editors, *Lecture Notes in Computer Science*, volume 3017, pages 371 – 388. Springer-Verlag Heidelberg, 2004.

[Sch90] C.P. Schnorr. Efficient identification and signature schemes from smart cards. In *Crypto '89*, volume 435, pages 239 – 251. Springer LNCS, 1990.