UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMPLEMENTACIÓN DE UN SISTEMA DE VOTACIÓN ELECTRÓNICA EFICIENTE PARA MÚLTIPLES CANDIDATOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

PAMELA CORDERO JURE

PROFESOR GUÍA:
SR. ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
SR. MARCOS KIWI KRAUSKOPF
SR. JOSÉ PIQUER GARDNER

SANTIAGO DE CHILE
ABRIL 2008

A mis padres y a Nicolás
por su inmenso cariño y entrega.

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN
POR: PAMELA CORDERO JURE
FECHA: 21 de ABRIL DE 2008
PROF. GUÍA: SR. ALEJANDRO HEVIA ANGULO

IMPLEMENTACIÓN DE UN SISTEMA DE VOTACIÓN ELECTRÓNICA EFICIENTE PARA MÚLTIPLES CANDIDATOS

Un área bastante estudiada dentro de la criptografía es la votación electrónica. El interés por ésta radica en que es potencialmente menos susceptible a errores que la votación tradicional en papel. En particular, se elimina el error humano (por ejemplo, anulación del voto por anotación deficiente) y errores en los resultados debido a escrutinios fallidos.

Muchos esquemas han sido propuestos en el estudio de la votación electrónica, los que difieren en las técnicas criptográficas utilizadas. Sin embargo, todos ellos tienen una finalidad en común: Mantener la privacidad del voto y al mismo tiempo garantizar el buen funcionamiento en cada uno de sus procesos. Es crucial que el sistema sea eficiente y que permita la elección entre múltiples candidatos. Aún no se encuentra implementado en Chile un esquema de votación electrónica de estas características.

La presente memoria está centrada en el estudio y la implementación de un esquema de votación electrónica eficiente, que permita al votante escoger entre múltiples candidatos. Además, se permite la votación por múltiples candidatos dentro de un mismo sufragio o papeleta de votación de acuerdo a las reglas previamente establecidas en el proceso electoral.

En la primera parte de este trabajo, se realiza un estudio de las herramientas criptográficas que serán utilizadas, incluyendo entre ellos algunos conceptos básicos de teoría de números, asi como de esquemas de encriptación y firmas digitales. Luego, se estudia el esquema implementado en cuestión y el detalle de cada una de sus componentes. Finalmente, se proponen algunas extensiones para el esquema implementado, orientadas a mejorar la calidad del sistema y se detallan las dificultades que se pueden presentar al llevar este esquema de votación a la práctica.

Financiamiento: Pamela Cordero agradece el valioso apoyo financiero de Conicyt via el proyecto Fondecyt 1070332 del Prof. Alejandro Hevia.

AGRADECIMIENTOS

Este trabajo no habría sido posible sin el apoyo de mi profesor guía Alejandro Hevia. Le agradezco por su dedicación, sus enseñanzas y sus sabios consejos. Agradezco también a los profesores Marcos Kiwi y José Piquer, por sus muy acertadas recomendaciones para mejorar la calidad de este trabajo.

Agradezco también a mi novio Nicolás, por su paciencia y apoyo incondicional. Por estar siempre a mi lado, aun en tiempos difíciles.

Agradezco a mis amigos, y en especial a Felipe, con el cual compartí todos estos años y con quien azarosamente siempre hemos seguido caminos similares.

Por último, agradezco a mis padres, por su cariño, entrega, esfuerzo y ayuda. Gracias a ellos fue posible llegar al final de este camino.

Índice general

Capítulo 1

Introducción

Desde la antigüedad los sistemas de votación han sido necesarios para que exista una forma de expresar las opiniones individuales en forma democrática. El escoger a un líder, decidir sobre una acción a tomar, determinar si alguien debe o no abandonar un grupo, son algunos ejemplos en que la votación se hace necesaria. El fin de ello es tomar una decisión como grupo, y ya que siempre pueden existir distintas opiniones, rescatar la opinión que tenga mayor acogida.

Existen diversas formas de votar. Por ejemplo, en una votación el voto puede ser público, como es el caso la votación a brazo alzado y la votación por aclamación. En ellas, cualquier integrante puede saber cuál fue la elección de otro votante. O bien puede ser privado, como ocurre en la votación actual en papel en que el votante marca su elección en secreto y deposita su papeleta de votación en una urna. Asimismo, se pueden fijar inicialmente distintas reglas de elección. Con ellas se define la forma mediante la que se llegará a un resultado, por ejemplo, la regla de la mayoría o la mitad más uno.

La votación pública es una forma sencilla de llevar a cabo un proceso electoral, y al hablar de sencillez nos referimos a diversos factores, como por ejemplo, que no sea necesario resolver problemas como mantener la privacidad de un voto. Es sencillo para cualquier integrante verificar que los votos fueron contados correctamente ya que todos éstos son públicos. Sin embargo, la votación pública puede distorsionar los resultados que se esperarían en un escenario perfecto. La votación pública puede permitir situaciones no deseadas, como la presión social. Ésto ocurre, por ejemplo, con la venta de votos, coerción, etc. Es por esta razón que para elecciones en las cuales el resultado tenga gran impacto social, es necesaria la privacidad, para evitar problemas que induzcan al votante a emitir un voto contra su voluntad.

Garantizar privacidad en una votación, pero a la vez satisfacer aquellos beneficios con los que cuenta la votación pública es uno de los fines de este trabajo, que se busca alcanzar mediante el uso de votación electrónica. Aún no se encuentra implementado en Chile un sistema de votación que cumpla con estas características. En el mundo, existen diversas propuestas para esquemas de votación que reunen privacidad y verificación universal, muchas de ellas no han sido llevadas a la práctica, como es el caso de este trabajo. Se trabajó sobre

esquemas propuestos y a la vez se incluyó módulos adicionales para fortalecer su seguridad. A continuación, se dará una breve explicación de conceptos que serán claves para comprender mejor el trabajo realizado.

1.1. ¿Qué es un proceso electoral o votación?

Es transformar en una decisión la voluntad de los ciudadanos o de un grupo determinado con respecto a un ítem en particular. Para ésto, se recolecta la voluntad de la gente mediante votos. Por ejemplo, en un curso de colegio se quiere escoger al mejor compañero, para ésto, cada uno de los integrantes del curso vota por el compañero que desea que salga escogido y el ganador es aquel que obtiene la mayor cantidad de votos.

Un proceso electoral consiste en un conjunto de reglas que rigen la forma en que los votantes pueden expresar sus deseos y cómo éstos se combinan para llegar a un resultado final.

1.2. Procesos involucrados dentro de la votación

En una votación existen distintos procesos asociados, que se explican a continuación en el orden en que se llevan a cabo.

- Registro: Proceso previo a las elecciones, donde los votantes demuestran su identidad y derecho a voto. Se crea un rol electoral.
- Validación: Durante las elecciones, los votantes son autentificados antes de realizar su votación.
- Votación: Los votantes manifiestan su preferencia.
- Escrutinio o conteo: Cómputo de los votos al final del periodo de votación.

Existen varios tipos de votación, entre ellas la votación manual, o votación en papel que se utiliza en las elecciones presidenciales en muchos países y la votación electrónica, cuya aparición es reciente, y ya ha sido implementada en países como Inglaterra, Estonia, Suiza, Brasil, India, Estados Unidos, Alemania, entre otros.

1.3. ¿Qué es votación electrónica?

Son los procesos electorales realizados mediante con ayuda de máquinas electrónicas. Es decir, todos aquellos actos electorales factibles de ser llevados a cabo utilizando tecnologías

de la información. Se espera que un sistema de votación electrónica cumpla con las siguientes características:

- Privacidad: Se requiere mantener la privacidad del voto. Ésto es, que no haya relación entre un voto y un votante.
- Incoercibilidad: Significa que no haya forma de que un votante pueda demostrar a terceros por quién votó. Una variante de ésto es la venta de votos o el cohecho.
- Exactitud: Ningún voto puede ser alterado, duplicado o eliminado sin ser detectado.
- Democracia: Sólo pueden votar aquellos que tengan derecho a voto (miembros del registro electoral), y cada votante puede votar una y sólo una vez.
- Robustez: Todos los requerimientos de seguridad se deben satisfacer completamente, a pesar de fallas o mal comportamiento por cualquier coalición razonable de los participantes (votantes, autoridades, observadores externos).
- Practicabilidad: Compatibles con una gran variedad de plataformas, fácil de utilizar. La simplicidad es necesaria de modo que la instrucción del votante sea mínima, y así evitar posibles errores. El sistema puede ofrecer alternativas para quienes viajan y para quienes tienen problemas físicos de modo de no negarles el derecho a voto.
- Verificación Universal: Todo espectador puede asegurarse de que el proceso electoral es correcto y que el resultado de las elecciones está bien computado. Además, el proceso debe ser auditable en cada una de las etapas de su funcionamiento.

1.3.1. Tipos de votación electrónica

Existen dos tipos de votación electrónica: La votación presencial en ambiente controlado y la votación por Internet, también llamada votación no supervisada. Sus características son las siguientes:

- Votación presencial en ambiente controlado: En este tipo de votación electrónica el registro se hace por entidades autorizadas. La validación dependiendo del caso puede ser física (por encargados electorales) o electrónica (por algún medio de identificación digital). Los procesos de votación y escrutinio se hacen de manera electrónica. La votación se lleva a cabo en los recintos electorales. El voto se puede almacenar en un dispositivo DRE[1] o enviado directamente a una central por medio de una conexión a Internet segura.

[1] "Direct Recording Electronic". Son máquinas electrónicas diseñanas para la votación, donde las opciones o candidatos posibles son visibles al usuario en la pantalla. El votante genera su voto directamente mediante el uso de *touch-screen*, pulsadores o algún dispositivo similar. Las preferencias del votante se almacenan en estas máquinas directamente agregándolas a las preferencias del resto de los votantes.

- Votación por Internet o no supervisada (i-voting): En este caso el votante no es supervisado mientras vota, vale decir, puede estar en casa, en el trabajo, etc. El proceso de registro puede ser físico o electrónico con algún tipo de identificación digital. La validación, la votación y el escrutinio se realizan de manera electrónica. La votación por Internet requiere de un nivel de seguridad mucho mayor. Por ejemplo, se debe mantener a los usuarios lo suficientemente educados para que sus máquinas no se encuentren comprometidas al momento de la elección. Aun así, ésto último no siempre es suficiente para evitar ataques bajo la tecnología actual (por ejemplo, ataques de coerción).

1.4. Ventajas y desventajas de la votación electrónica

La votación electrónica tiene grandes ventajas sobre la votación manual, entre ellas se encuentran:

- Reducción en tiempos de ejecución de procesos tales como votación, verificación de votos y escrutinio; y un considerable ahorro social (el costo en tiempo y movilización en caso de i-voting, vocales de mesa en todos los casos).
- Se introduce un nuevo concepto llamado "Verificación Universal". Éste permite agregar un grado importante de confianza en la gente sobre los resultados de la votación.
- En el caso de la votación vía Internet, la votación electrónica facilitaría el acceso a más personas. Ésto permitiría que los resultados obtenidos en una votación representaran mejor al total de la población. Por ejemplo, hoy cerca de un 30 % de la población potencialmente votante en Chile no participa en las elecciones.
- La votación electrónica promete reducir las tasas de errores (*undervoting* o blancos, *overvoting* o nulos) que se tienen actualmente de manera significativa.

Sin embargo, una desventaja de la votación electrónica sobre la votación manual es la familiaridad o costumbre del votante en el proceso. Esta desventaja podría ser reducida con el tiempo, a través de educación. Por ejemplo, se podría instruir a los votantes para que entiendan sus propiedades. Otras desventajas de un sistema de votación electrónica son:

- La votación manual aún es considerada segura en la percepción ciudadana.
- El ciudadano promedio está acostumbrado al proceso de votación manual el cual le es bastante intuitivo.
- Una mínima duda en un votante respecto a la privacidad de su voto en un futuro próximo puede involucrarlo en un ataque de coerción. La votación vía Internet es más susceptible a este tipo de ataques.
- Los datos digitales suelen ser más fáciles de ser alterados o destruidos que los datos físicos en gran escala.

- Muchos sistemas electrónicos pueden verse afectados por ataques tales como aquellos de denegación de servicios.
- La votación vía Internet (si algún día es factible) sólo podrá volverse democráticamente aceptable cuando todos los usuarios tengan acceso a Internet.
- Se requiere una gran inversión inicial.

La votación electrónica involucra muchos factores importantes. Estos factores son: criptografía, seguridad de redes y seguridad de *software*. Más aún, se introduce un nuevo concepto que es el de "Verificación Universal", en otras palabras, que cualquier persona, sea votante, autoridad o un tercero, puede estar seguro que todo proceso en una votación se está llevando a cabo correctamente y que cualquier alteración será detectada. Es más, se debe desligar la relación entre un voto y un votante. Se debe asegurar que el voto fue incorporado al conteo correctamente sin decir quien lo envió. También, se espera que el voto al encontrarse encriptado bajo altos niveles de seguridad garantice que el contenido de éste no será revelado, sólo será descartado bajo las reglas preacordadas (legales) del proceso de votación.

1.5. Votación electrónica en la práctica

A lo largo de la historia la votación ha ido evolucionando. Parte de esta evolución se ha reflejado en los siguientes sistemas:

1. **Votación en la antigüedad:** La votación existe desde que existe democracia. Este término proviene de la antigua Grecia. Su significado original era "el poder del pueblo" (demo = pueblo y kratos= poder). Este término nació en la ciudad de Atenas donde se estableció una forma de gobierno en que la ciudadanía jugaba un rol relevante. Las autoridades representaban al pueblo, siendo la mayoría escogidas por sorteo y las más importantes se escogían mediante aclamación popular. De esta manera el pueblo participaba de manera activa en las decisiones fundamentales que se tomaban en la ciudad.

   En este tipo de votación sólo se incluía a los hombres que superaban cierta edad. No era considerada la opinión de mujeres y niños, ni tampoco la de los esclavos. Tampoco consideraba a todo el pueblo sino a unos pocos. Pero sí era un inicio de democracia que ha crecido hasta la fecha, considerando a la población sin diferencias sociales, ni de sexo. Además, este tipo de votación era aún pública. Con el tiempo se fue requiriendo la privacidad del voto, por lo tanto, los tipos de votación fueron evolucionando.

2. **Voto australiano:** Corresponde al clásico voto secreto en papel llamado también *Mark-Choice Ballot*, en él se listan los nombres de todos los candidatos. Los formularios de votación son impresos en serie con un número identificador para cada uno. El votante marca su elección en privado. Sin embargo, este modo de votación no impide que los votos puedan ser robados, destruidos o incluso sustituidos.

3. **Máquinas de votación mediante palancas mecánicas:** Las máquinas de votación mediante palancas fueron inventadas en el año 1982 con el fin de eliminar algunas manipulaciones de los votos que ocurrían en sistemas de votación basados en papel. Para votar en ellas, el votante ingresaba a la sala de votación donde se encontraba una de estas máquinas. Esta máquina tenía una palanca cercana al nombre de cada candidato. El votante debía mover la palanca del candidato escogido. Estas máquinas eliminaban el conteo manual de votos, pero de todas formas, aún se podía manipular el total alterando la máquina, por ejemplo, para que ésta no contara algunos votos.

4. **Conteo asistido por computador:** Entre ellos se encuentran los *Punchcard System* (1964) y los *Optical Scan* (1980). En ambos casos el formulario de votación es almacenado en un lector electrónico que cuenta los votos y los almacena en la memoria de un computador. Ésto hace que las manipulaciones sean más difíciles que en el caso manual. Pero no obstante, aún podrían existir alteraciones a nivel de *hardware* y *software*.

5. **Máquinas de votación electrónicas (DRE):** Fueron creadas en 1970, como una forma más sofisticada de votación que las máquinas de votación por medio de palancas mecánicas. Los votantes realizan la elección de sus candidatos mediante la pulsación de botones, *touchscreen* o utilizando dispositivos similares, por ejemplo, dispositivos de reconocimiento de voz.

1.5.1. DRE (o votación por computador): Ventajas y problemas

Las DRE ofrecen muchas ventajas por sobre otros sistemas de votación electrónica. Por ejemplo, permiten notificar al votante advirtiéndolo antes de votar nulo o blanco. Es decir, pueden advertir antes de que el voto sea almacenado para no incurrir en errores. Incluso, las DRE son capaces de satisfacer ciertos requerimientos de accesibilidad[2] y aquellos para la prevención y corrección de errores, como por ejemplo, caidas del sistema. Con el uso de ellas se reducen algunos riesgos que existen en el voto en papel, como por ejemplo, la eliminación de ciertos votos durante el escrutinio. Además, se pueden reducir considerablemente el número de votos perdidos por errores de los votantes. Sin embargo, existen muchas discusiones respecto a su uso en la práctica y los problemas que pueden haber. Éstos se discuten a continuación.

1. **Confianza:** En estas máquinas, a diferencia del voto en papel, el votante ve sólo una representación del voto que es almacenado electrónicamente. Este problema es similar al descrito en las máquinas con palancas. En caso de existir alguna modificación interna del voto, ni el votante ni el observador podrían saberlo. El votante deberá confiar en la integridad del sistema que está utilizando si es que va a confiar en el resultado de las elecciones y en la legitimidad de los gobiernos formados como consecuencia de ello.

   Actualmente son muy pocas las empresas que fabrican DRE. En ellas se concentra el mercado y, por lo tanto, ésto puede significar un problema. Expertos consideran que es mucho más difícil hacer un ataque generalizado exitoso si existen muchos sistemas diferentes que deben ser comprometidos.

[2]Para permitir el voto de personas con ciertas discapacidades, por ejemplo no videntes.

2. **Ataques y vulnerabilidades:** En la mayoría de los sistemas computacionales un ataque siempre es posible si el adversario cuenta con recursos suficientes. En la práctica la seguridad de estos sistemas se basa en hacer que los recursos necesarios para quebrar el sistema sean inalcanzables por un atacante en el mundo real. En el desarrollo de un sistema de votación electrónica es crucial garantizar la implementación de un programa donde explotar vulnerabilidades es infactible.

   Los programas maliciosos suelen ser escritos de manera tal que son muy difíciles de identificar, más aún, la complejidad de los programas en una DRE hace más difícil detectar modificaciones no autorizadas y, por lo tanto, los hacen más vulnerables. Por ejemplo, un *hacker* malicioso pudiera insertar código dañino dentro de un sistema de votación electrónica. Tal programa pudiera ir rechazando o cambiando votos de manera aparentemente aleatoria o podría adaptativamente cambiar su comportamiento de acuerdo al camino que va tomando el resultado (estas dos formas anteriores son muy difíciles de reconocer).

   En la votación se deben considerar distintos tipos de ataques y atacantes, de acuerdo al grado de impacto que provoca. El mejor atacante es aquel que es capaz de modificar el total de los votos. Se considera un atacante como tal, si éste es capaz de modificar los resultados de una elección. Por ejemplo, si consideramos un adversario que sólo es capaz de modificar un voto o afectar a una máquina, es distinto a uno que es capaz de afectarlas a todas. Debe tenerse en cuenta, también, que el efecto de un ataque es más significativo si el apoyo popular a ambos candidatos es similiar.

   Se identifican distintos tipos de vulnerabilidades, vale decir, debilidades en el sistema que podrían ser explotadas por un atacante. Existen las de tipo técnicas y las sociales.

   - Entre las vulnerabilidades técnicas que se pueden encontrar están aquellas que comprometen al código. Mientras más complejo es un *software* más vulnerable es éste a ataques. Expertos argumentan que es imposible anticiparse a todas las vulnerabilidades y tipos de ataques en *software* complejos. Las DRE son máquinas que contienen *software* complejos. Es por ello que éstas requieren de un muy buen diseño para disminuir riesgos de ataques, como por ejemplo, ataques de tipo *bufferoverflows* o DOS (*denial of service* o denegación de servicio). Si bien la criptografía es uno de los elementos más fuertes para garantizar seguridad, un mal uso de ésta podría introducir vulnerabilidades en el sistema. Por ejemplo, un sistema que encripta utilizando una clave privada conocida no es seguro a pesar de que utilice fuertes modelos criptográficos.

     Otro tipo de vulnerabilidades técnicas son aquellas que involucran la conexión a otros computadores. Aquellos con conexión directa a Internet pueden ser explotados por *hackers* maliciosos. Es por ésta y otras razones que la votación por Internet es tan poco atractiva. Por lo tanto, frecuentemente se recomienda que en un sistema de tipo DRE, generadores y contadores de votos no se encuentren conectados a Internet (a esta práctica se le llama *air gapping*). El traslado de la información en estos casos se hace por CDs, *memory cards*, etc. Sin embargo, es posible, si existe autentificación, minimizar la existencia de vulnerabilidades en el traspaso por Internet utilizando funcionalidad reducida. También se debe considerar la forma en que el votante interactúa con las DRE. En Estados Unidos uno de sus sistemas hacía ésto mediante *smartcards* mal diseñadas, las cuales podían

ser falsificadas por los mismos votantes para votar repetidas veces.

- Dentro de las vulnerabilidades sociales existen las de ingeniería social, basadas en cómo el usuario interactúa con el sistema. Buenas políticas y prácticas de seguridad pueden prevenir algunos de los ataques que utilizan estas vulnerabilidades. El personal que forma parte de las elecciones es clave. Ellos son quienes deben llevar a cabo las políticas y procedimientos de seguridad, por lo tanto, deben ser entrenados para ello, de lo contrario, podrían ser vulnerables a ataques de ingeniería social.

3. **Uso de código propietario:** El uso de código fuente propietario impide que expertos ajenos a la empresa puedan evaluar vulnerabilidades en el sistema. Debido a las complejidades en el código, esta medida hace que sea aun más difícil garantizar que el código se encuentra libre de programas maliciosos. Algunos expertos favorecen el uso de *open source* o código abierto para aumentar la seguridad en un sistema, dado que se podrían encontrar más fallas de seguridad de las que es posible encontrar en código propietario. Aquellos que apoyan el código propietario argumentan que al ser éste privado hace que las fallas sean más difíciles de encontrar y, por lo tanto, las hace más difíciles de ser explotadas. A ésto se le denomina *security through obscurity*, y es usualmente considerado una medida de seguridad frágil porque una vez que este código sale a la luz el daño es irreparable ya que éste no podrá hacerse secreto nuevamente[3].

Consideraciones generales:

Para proteger un sistema se deben establecer medidas de protección, detección y reacción. Las medidas de protección sirven para que un ataque sea muy difícil de llevar a cabo exitosamente. Las medidas de detección y reacción se deben establecer pensando en un peor caso. Es mejor asumir que un atacante podría resultar exitoso y tener medidas para detectarlo y detenerlo a tiempo, para que su daño sea mínimo. Para detectar fallas, por ejemplo, se podría utilizar un *log* que registre toda acción que se lleve a cabo en la máquina.

Existen tres elementos claves en una votación para reducir vulnerabilidades. Uno de estos elementos es el personal, el cual debe estar capacitado para saber reaccionar ante los problemas que se presenten. El segundo elemento es la tecnología, la cual debe involucrar un buen diseño a nivel de *software* y *hardware*. Y por último, deben existir buenas políticas y procedimientos de seguridad. Si el enfoque no se encuentra bien balanceado entre estos tres componentes se pueden generar vulnerabilidades.

Es también una buena medida establecer estándares de votación. Dada la precocupación que han causado los problemas mencionados anteriormente, existen estándares establecidos en algunos países que implementan votación electrónica. Por ejemplo, en Estados Unidos, para la votación asistida por computador se introdujo en 1990 el estándar llamado VSS (Voting System Standards). Junto a estos estándares fue desarrollado y administrado por NASED[4]

[3]En el año 2003 se descubrió un servidor web público que contenía parte del código fuente de *Diebold*, uno de los mayores vendedores de sistemas de votación. Se descubrió que éste tenía muchos problemas que podrían permitir que los votos fueran manipulados por distintas personas, por ejemplo, los votantes, trabajadores de las elecciones, *hackers* en Internet, e incluso desarrolladores de *software*.

[4]Asociación Nacional de Directores de Elecciones Estatales.

un programa de testeo y certificación, en el que una autoridad de testeo independiente (ITA) escogida por NASED testea el sistema de votación y certifica si cumple o no con los estándares VSS a nivel de *hardware* y *software*. El código certificado es guardado por la ITA, y si ocurre alguna duda respecto a alguna máquina el código se compara con el certificado por la ITA. Los sistemas que han recibido certificación de NASED, además, deben pasar por procesos de certificación a nivel de estado y local antes de ser utilizados en los procesos electorales. La adopción de estos estándares en algunos estados ayudó a disminuir problemas como los mencionados anteriormente, aunque algunos cuestionan la efectividad de la certificación en términos de mejorar la seguridad.

Los estándares son importantes en la seguridad dado que especifican atributos medibles que un sistema necesita para ser considerado confiable. Sin embargo, no es posible anticiparse a todas las formas en que un sistema podría ser atacado, por lo tanto, ésta no puede considerarse como una única medida de seguridad.

1.5.2. Voter verifiable secret ballot

Algunos observadores creen que los principales problemas de seguridad asociados a las lagunas de transparencia y observación entre el proceso de votación y escrutinio con las DRE no pueden ser resueltos con el uso de procedimientos de seguridad, estándares, certificación y testeo. Concuerdan en que el único enfoque confiable es utilizar votos que los votantes puedan verificar de manera independiente a las DRE y que estos votos deben transformarse en los oficiales ante cualquier necesidad de recuento. Una técnica para hacer ésto es el uso de los llamados *voter verifiable secret ballot*. En este caso el DRE imprime un voto en papel con las opciones escogidas con el votante. Si el votante está de acuerdo con lo impreso el voto es almacenado en una *ballot box* o urna. La idea no es contar todos los votos al final de las elecciones, sino tener un respaldo ante dudas respecto al buen funcionamiento de una determinada máquina, por ejemplo, si un gran porcentaje de las personas que ha utilizado una máquina se ha quejado. El uso de esta práctica tiene fuertes ventajas, pero a la vez, existen críticas respecto a la necesidad de ello.

Ventajas:

- Todo recuento estará basado en un registro independiente (voto en papel) el cual fue verificado previamente por el votante.
- Toda elección puede ser auditada, y en caso de existir dudas respecto al funcionamiento de una máquina o en caso de existir repetidas fallas en una máquina se podrá hacer un recuento en papel de los votos que en ella se hayan realizado.
- Este método ayuda al votante a confiar en la legitimidad de los resultados de las elecciones. La razón de ésto, es que los votantes sabrán que los votos que verificaron estarán disponibles para un recuento.

Críticas:

- Hace el proceso más complejo y consume mucho más tiempo del votante
- El uso de impresoras incrementa el costo de administración de una elección y el riesgo de fallos mecánicos de una máquina de votación.
- El conteo a mano consume más tiempo y puede producir más errores que el conteo a máquina, sin embargo, se puede utilizar sólo en caso de dudas (auditorías).

1.6. Votación electrónica usando técnicas criptográficas

En votación electrónica es necesario el uso de técnicas criptográficas para poder así cumplir con las propiedades de verificación universal. A continuación, se introducirán algunos conceptos que serán utilizados más adelante.

1.6.1. Criptografía de clave pública

La criptografía de clave pública se basa en el uso de un par de claves diferentes relacionadas, una de las cuales es pública y la otra privada. Se utilizan tanto para la encriptación (privacidad de datos) como para la generación de firmas digitales (autentificación de mensajes). A continuación, se verá la aplicación en cada uno de los casos.

1. **Encriptación:** En estos esquemas una de las claves es la encargada de encriptar el mensaje, y es de conocimiento público. Por ejemplo, ésta puede encontrarse en un repositorio de claves públicas. La segunda clave corresponde a la clave privada, la cual es de conocimiento único y exclusivo del receptor. Con la clave privada es posible obtener el mensaje en texto plano. En otras palabras, cualquier persona es capaz de encriptar un mensaje para un destinatario Y pero sólo Y puede desencriptar el mensaje.

   En estos esquemas de encriptación, llamados esquemas de encriptación asimétrica, a diferencia de los esquemas de encriptación simétrica que utilizan una clave en común entre el emisor y el receptor, no se debe acordar previamente una clave entre ambos participantes. Es el receptor quien genera su par de claves pública y privada y comparte su clave pública con el resto de los emisores. Por otro lado, cada receptor necesita sólo un par de claves para recibir mensajes de cualquier emisor, a diferencia de los esquemas de clave simétrica en los cuales debe existir una clave para cada par *(Emisor, Receptor)*

   Suena lógico preguntarse cómo es que no se puede obtener la clave pública a partir de una clave privada. Sin embargo, este proceso puede modelarse matemáticamente como una función unidireccional [1], es decir, como una función fácil de calcular en un sentido,

pero cuya inversa es difícil de calcular. El cifrado de clave pública se basa típicamente en problemas intratables de teoría de números.

Quien envía el mensaje debe estar seguro de tener la clave pública correspondiente, ya que si no lo está, el mensaje podría ser encriptado, por ejemplo, para un tercero malicioso. Para dar garantía de que una clave pública pertenece a alguien en particular, existen las Autoridades Certificadoras. Ellas, mediante un certifiado digital (firma) garantizan que una clave pública pertenece a alguien determinado.

La encriptación de clave pública es teóricamente quebrantable al igual que la encriptación de clave simétrica. Ahora bien, en la práctica se usan claves tan grandes que el tiempo necesario para quebrarla (recuperar información del mensaje sin la clave privada) es de cientos de años. Por consiguiente, si las claves son más extensas, son más seguras.

2. **Firmas Digitales:** Las firmas digitales son métodos criptográficos que se utilizan para asociar la identidad del firmante a un mensaje o documento. Es decir, tal como ocurre con las firmas convencionales, las firmas digitales pueden utilizarse, por ejemplo, para víncular un mensaje a quien lo envió, o para la aprobación de un documento digital. Para ésto, es el firmante quien genera el par de claves, una pública y una privada. Éste utiliza su clave privada para generar la firma digital. Todo aquel que desee verificar la firma sólo necesita la clave pública. La clave pública, al igual como ocurre en la encriptación asimétrica, es de conocimiento público y puede encontrarse en un repositorio de claves.

Algunos ejemplos de esquemas de encriptación que utilizan criptografía de clave pública son ElGamal y RSA (ver secciones 2.2.1, 2.2.2). Entre los esquemas de firmas se encuentran, por ejemplo, RSA (ver sección 2.3.1), DSA y ElGamal, entre otros.

1.6.2. Zero-knowledge proofs

Las *zero-knowledge proofs* o ZKP son métodos interactivos mediante los que una entidad (*prover*) quiere demostrar a otra entidad (*verifier*) que una cierta afirmación (matemática) es correcta sin revelar otra cosa más que la veracidad de la afirmación. Para ésto, se utilizan pruebas interactivas en las que el *verifier* mediante desafíos se convence de que la afirmación del *prover* es cierta. Una forma sencilla de entenderlo es mediante el ejemplo de la figura 1.1.

Inicialmente Alicia desea demostrar a Bob que conoce una forma de abrir la puerta que se encuentra al interior del laberinto. Bob la pone a prueba y para ésto le pide a Alicia que ingrese al laberinto por el camino que ella prefiera (Bob desconoce su elección). Una vez que Alicia se encuentra dentro del laberinto Bob la desafía a salir por el lado que él escoja. Una vez terminado este proceso, Bob desconoce qué recorrido siguió Alicia dentro del laberinto. Sin embargo, él puede estar convencido que con un 50 % de probabilidad Alicia conoce una forma de abrir esa puerta. Si Alicia y Bob realizan este experimento repetidas veces la probabilidad de que Alicia diga la verdad irá creciendo. Con una cierta cantidad de iteraciones Bob podrá convencerse de lo que Alicia afirmaba saber, pero de todos modos, desconocerá la

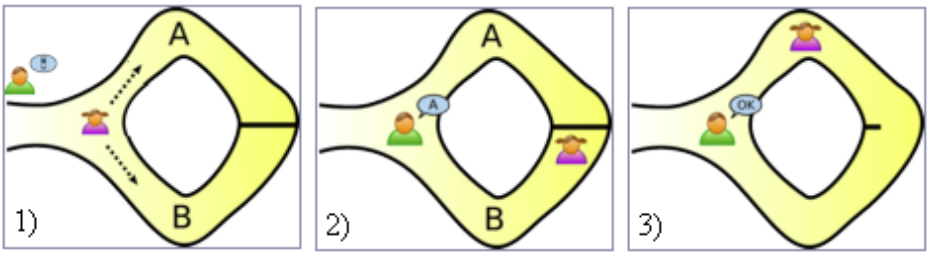

Figura 1.1: Ejemplo sencillo de una zero-knowledge proof

forma que tiene Alicia para abrir aquella puerta. Más aún, Bob no podrá utilizar la interacción con Alicia para demostrar a otra persona (Charly) que Alicia efectivamente puede abrir la puerta. Ésto porque Charly nunca podrá estar seguro que existió un acuerdo de antemano entre Alicia y Bob. La consecuencia es, por lo tanto, que aunque Bob se pueda convencer de la afirmación de Alicia, éste no ha "aprendido" nada, ya que aún no puede convencer a un tercero (Charly).

Uno de los usos que se da a las ZKP en protocolos que utilizan criptografía es forzar un comportamiento honesto mientras, por otro lado, se mantiene la privacidad. Para más detalles ver sección 2.5.

1.6.3. Secret sharing

Como su nombre lo dice, los *secret sharing* son métodos para distribuir un secreto s entre un número n de participantes. Cada uno de los n participantes posee un *share* o trozo del secreto. Sin embargo, cada participante por sí sólo no puede utilizar su *share* para obtener información alguna acerca del secreto, pero un subconjunto de $t \leq n$ participantes sí pueden reconstruirlo.

Inicialmente los participantes pueden necesitar de un *dealer* o entidad confiable la cual se encarga de generar y distribuir el secreto entre ellos. Dependiendo del caso, s puede ser generado de manera distribuida entre los participantes. De esta manera, no se hace necesario depositar la confianza en sólo una entidad (*dealer*) y, finalmente, al igual que en el caso anterior, cada participante obtiene su trozo del secreto.

Uno de los esquemas más conocidos es el de Shamir [2] el cual es explicado en la sección 2.7.

1.6.4. Modelos criptográficos utilizados en votación electrónica

La criptografía juega un rol muy importante en la seguridad de la votación electrónica. Para integrar seguridad se pueden emplear varios tipos de soluciones criptográficas. Los

estudiados son los siguientes:

- Modelo *Mix-Net*: alternativa criptográfica a un canal anónimo. Utiliza la primera propiedad de ElGamal, es decir, la reencriptación pública (ver sección 2.2.1). Permite desligar el mensaje encriptado de quien lo envía, de manera que el receptor no sepa de quién vino, y que un observador externo tampoco pueda saberlo. Se compone de varios servidores llamados *mixes*. Cada *mix* toma un conjunto de mensajes encriptados, los aleatoriza (reencriptando), luego entrega como salida un conjunto de mensajes permutados, tal que la salida y la entrada no son vinculables. Por ejemplo, si cada mensaje representa un voto, entonces el conjunto de mensajes encriptados final es equivalente al anterior (el conjunto de mensajes o votos es el mismo), pero con los votos representados con una encriptación diferente y con otro orden. Dos propiedades destacables de estos sistemas, son la verificación universal y la eficiencia. Además, basta con que una *mix* actúe correctamente para poder asegurar la privacidad de los votos.

- Modelo de Firmas Ciegas: Este modelo se caracteriza porque el firmante no "ve" lo que firma. Gracias a ésto, se puede verificar el voto sin sacrificar la privacidad. La confianza en que se esté firmando lo adecuado debe ser adquirida por el firmante por otros medios complementarios (*zero-knowledge* [1, 3]). Se caracteriza por su pequeño tiempo de cómputo, incluso en grandes elecciones. Requiere, también, de canales anónimos.

- Modelo de Benaloh: Usa el esquema de *"Homomorphic Secret Sharing"* [2, 4] o compartición de secretos homomórfica. Cada votante comparte su voto con $n \geq 2$ autoridades de voto. Al final del período de votación cada autoridad combina todos los trozos de votos recibidos para conseguir un trozo del total de los votos. Finalmente, las autoridades combinan sus trozos para conseguir el total de votos. Para darle robustez, al menos t de las n ($t \leq n$) autoridades deben combinar sus trozos para recuperar el total.

- Modelo de Encriptación Homomórfica: Este modelo usa un esquema de encriptación homomórfica como ElGamal [5] o Paillier [6]. Su aporte es que nadie puede saber por quién ha votado cada votante, dado que los votos se contabilizan directamente sin ser desencriptados. Ésto es porque un esquema de encriptación homomórfica tiene una propiedad en que la multiplicación de los textos cifrados equivale a la suma de los textos planos ($E(a) \cdot E(b) = E(a + b)$).

1.7. Descripción general del esquema de votación propuesto

En el presente trabajo se abordará el problema considerando un esquema de encriptación en el que los votos son encriptados utilizando una clave pública. Un conjunto de tres autoridades serán las responsables de generar los parámetros iniciales y el par de claves necesarias para la encriptación de los votos. Se podrá verificar en todo momento mediante pruebas generadas que las autoridades estén actuando correctamente y el sistema funcionará con normalidad a pesar que una de ellas resultara corrupta. Los votantes al manifestar su preferencia

enviarán no sólo su elección o preferencia encriptada, sino, además, una *zero-knowledge proof* de aquella encriptación y una firma digital de los dos valores anteriores. Las autoridades se encargarán, una vez finalizado el proceso de votación, de verificar la validez de los votos. En base a eso, considerarán dentro del total aquellos votos encriptados cuyas papeletas sean aceptadas, ésto es, aquellos votos válidos y cuya firma es correcta. Posteriormente, un subconjunto de las autoridades utilizando *secret sharing* serán las encargadas de desencriptar el total de los votos que corresponderá a un único texto cifrado indicando la cantidad de votos para cada candidato, dadas las carácterísticas del modelo de encriptación homomórfico.

1.8. Motivación

La implementación de un esquema de votación electrónica de las características mencionadas anteriormente responde a una necesidad aún no satisfecha por los actuales sistemas de votación en Chile. Esta necesidad es permitir al votante y a todo observador tener certeza de que los procesos internos se están llevando a cabo correctamente. En el caso del votante, garantizar que su voto fue contabilizado y si este no lo fue, dar los motivos pertinentes. Todo lo anterior es un problema complejo si a ello agregamos la necesidad de la privacidad en el voto. Este problema puede ser resuelto mediante el uso del modelo propuesto en este trabajo.

Los sistemas actuales aún son susceptibles a ciertos tipos de corrupción, como modificación de resultados y anulación de votos. Con dicho esquema se podría considerar incluso la corrupción de alguna de las partes involucradas (servidores). Esta cualidad fortalece al sistema y dificulta cualquier tipo de tarea deshonesta a todo nivel (sea votante, observador o autoridad).

1.9. Sistemas de votación electrónica existentes

A continuación, se mencionarán algunos esquemas de votación electrónica que ya han sido implementados.

Uno de los sistemas donde se concentra gran parte del mercado es *Diebold*, sin embargo, también es uno de los criticados más fuertemente. Es un sistema comercial, su código es propietario y no posee verificación universal. Sumado a ésto, en el año 2002 parte del código de *Diebold* fue filtrado, y se descubrió un mal uso de la criptografía y diversas vulnerabilidades [7, 8].

Otros sistemas ya implementados que utilizan criptografía son: *VoteHere VHTi* [9], ésta es una implementación comercial cuyo código es propietario. Ciertos módulos son públicos para permitir la verificación durante el escrutinio. Utiliza *mix-nets*. *SureVote*, es un sistema comercial, el cual utiliza *mix-nets* y, basado en el esquema propuesto por Chaum [10], incorpora una componente de verificación por medio de imágenes. *E-vote* [11] utiliza encriptación homomórfica de Paillier [6] y su código es cerrado. *Sensus* [12] y *EVOX* [13] son

implementaciones basadas en firmas ciegas. *Sensus* tiene su código disponible bajo licencia.

Por último, es relevante mencionar el esquema propuesto en [14]. Éste utiliza encriptación homomórfica (ElGammal) y sus claves son generadas de manera distribuida, por lo tanto, la desencriptación se hace también de esta manera. Algunas diferencias entre [14] y el esquema escogido en este trabajo son: Por un lado, se encuentra el modo en que cada sistema considera un voto, en [14] se debe realizar ρ encriptaciones si el votante escoge ρ candidatos, y no una representación de ellos en conjunto (que es lo que se hace en el presente trabajo). Por otro lado, en [14] existen procesos en los que aún no se está llevando a cabo la verificación universal (por ejemplo, generación de claves).

1.10. Objetivos

1.11. Objetivos generales

Implementar un esquema eficiente de votación electrónica utilizando encriptación homomórfica. Éste deberá permitir la votación para múltiples candidatos y respetar las propiedades esperadas para un sistema de este tipo mencionadas anteriormente. Se implementarán los procesos de votación y escrutinio. Los procesos de registro y validación dependerán del tipo de proceso electoral.

1.12. Objetivos específicos

- Implementar un esquema de votación electrónica basado en encriptación homomórfica.
- Garantizar que la solución sea "Universalmente Verificable".
- Garantizar que la solución no revela información que vincule a un voto y un votante.
- Garantizar que todo voto válido será sumado al total, y si este paso no se realiza será detectado.
- Desarrollar la solución para votación presencial o *polling place voting*. Ésto debido a las claras desventajas de implementar *i-voting* en nuestros días, en cuanto a coerción.
- La votación electrónica conlleva una serie de cálculos matemáticos (exponenciaciones). La reducción en cantidad de este tipo de operaciones permite lograr las características mencionadas, pero consumiendo una menor cantidad de recursos (tiempo). En este trabajo se utilizarán esquemas que minimicen este tipo de operaciones.

Capítulo 2

Herramientas criptográficas

Durante el desarrollo de este trabajo se investigó las ventajas de la implementación de un esquema de votación electrónica y la manera de abordar el problema criptográficamente para garantizar privacidad e integridad. Se estudió en profundidad esquemas propuestos hasta la fecha [15, 16, 17], rescatando cualidades de ellos, para llegar así al esquema que finalmente fue implementado. Luego de analizar en profundidad estos tres esquemas, los cuales en general mantienen una estructura similar, se decidió implementar el esquema propuesto en [15]. Son dos las principales razones que hacen a este esquema más interesante. En primer lugar se destacan las ZKP, que si bien son más complejas que en los otros dos casos [16, 17] son mucho más eficientes. En segundo lugar, este esquema deja abiertos ciertos aspectos al estudio del implementador, como son la generación de claves y la generación de parámetros. Esta característica lo hace más interesante de ser implementado. Un último aspecto que tuvo influencia a la hora de decidir si aceptar o rechazar el esquema propuesto en [16] fue, que si bien este esquema dice estar enfocado a múltiples candidatos, el enfoque que se da es distinto al enfoque que se da en esta memoria. En [16] se habla de múltiples candidatos al tener más de dos opciones para elegir, pero no al poder elegir más de un candidato en una misma papeleta de votación, que es uno de los objetivos de este trabajo. Además, si bien este esquema podía extenderse para el caso de múltiples elecciones en una misma papeleta, ésto hacía que las ZKP, que eran un punto fuerte de esta solución, perdían su simplicidad y eficiencia. Ésto es, porque se debían calcular muchos valores que en la práctica disminuirían sustancialmente la eficiencia del sistema. Es por estas razones que se decidió, finalmente, implementar el esquema propuesto en [15], dado que aún no existen implementaciones disponibles de éste.

Por otro lado, se investigó sobre la factibilidad técnica, considerando que este sistema de votación electrónica se centrará en *Polling Place Voting*. Se evaluó algunas técnicas propuestas para las DRE en votación electrónica para garantizar de manera eficiente que la implementación se encuentra libre de código malicioso. Al hablar de eficiencia nos referimos a reducir al máximo la cantidad de líneas de código que deberán ser verificadas para garantizar el buen funcionamiento del sistema.

2.1. Conceptos básicos: Teoría de números

A continuación se explicarán algunos conceptos básicos de teoría de números que se utilizarán en el desarrollo de este trabajo.

2.1.1. Enteros mod N

Sean a, N enteros, donde $N > 0$. Existen enteros únicos $r < N$, q, tales que $a = Nq + r$. El entero r corresponde al resto que se obtiene al dividir a y N y es el resultado de la operación $a \bmod N$. A esta operación se le llama módulo N.

Se puede asociar a cualquier entero N el siguiente par de conjuntos:

- $Z_N = \{0, 1, ..., N-1\}$
- $Z_N^* = \{i \in Z : 1 \leq i \leq N \wedge mcd(i, N) = 1\}$

El primer conjunto es llamado el conjunto de enteros mod N. El segundo es el conjunto de todos los primos relativos de N, es decir, todo entero i cuyo máximo común divisor con N sea igual a 1.

2.1.2. Grupos

Sea G un conjunto distinto de vacío y sea $\cdot$ una operación binaria en G. Decimos que G es un grupo si satisface las siguientes propiedades:

- Clausura: Sea $a, b \in G$, entonces $a \cdot b \in G$.
- Asociatividad: Sea $a, b, c \in G$, entonces $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- Identidad: Existe un elemento $1 \in G$ tal que $a \cdot 1 = 1 \cdot a = a$ para todo $a \in G$.
- Inversibilidad: Para todo $a \in G$ existe un único $b \in G$ tal que $a \cdot b = b \cdot a = 1$.

Sea N un entero positivo, entonces Z_N es un grupo bajo suma módulo N y Z_N^* es un grupo bajo multiplicación módulo N [18]. Se llama orden en teoría de grupos al tamaño o cantidad de elementos de un grupo. Sea G un grupo. A un elemento $a \in G$ se le llama *cuadrado* o residuo cuadrático si tiene una raíz cuadrada $b \in G$ tal que $b^2 = a$.

2.1.3. Generadores y grupos cíclicos

Sea G un grupo, sea 1 su identidad y m su orden. Sea $g \in G$ un elemento cualquiera del grupo. Se le llama orden de g al primer entero $n > 0$ tal que $g^n = 1$. Denotamos al conjunto generado por g como:

$$< g >= \{g^i : i \in Z_n\} = \{g^0, g^1, ..., g^{n-1}\}$$

Este conjunto es, también, un subgrupo de G y su orden es igual al orden de g. Un grupo G es llamado cíclico si existe $g \in G$ tal que $< g >= G$. Por otro lado, un elemento $g \in G$ es llamado generador de G si $< g >= G$. Si g es generador de G entonces para todo $a \in G$ existe un único entero $i \in Z_m$ tal que $g^i = a$. Al entero i se le llama el logaritmo discreto de a en base g y lo denotamos como $DLog_{G,g}(a)$.

2.1.4. Problema DDH (Decisional Diffie-Hellman)

Sea G un grupo cíclico, g su generador y m su orden. Para este problema se consideran dos mundos. Se tiene un adversario el cual recibe entradas X, Y, Z. En ambos mundos X, Y son escogidos de manera aleatoria en G, pero la manera en que Z es escogido depende del mundo en el que se esté. En el mundo 1, $Z = g^{xy}$ donde $x = DLog_{G,g}(X)$ e $y = DLog_{G,g}(Y)$. En el mundo 0, Z es escogido de manera aleatoria en G. Se denotará por $a \in_R G$ a la acción de escoger un elemento de G en forma aleatoria (uniforme) y asignárselo a la variable a.

Sea A el algoritmo que retorna un bit b. El desafío para el adversario es determinar en qué mundo se encuentra. Para ésto, existen dos experimentos donde x e y son elementos escogidos de manera aleatoria en G:

- Experimento en mundo 1:

$$\begin{aligned} Exp_{G,g}^{ddh-1}(A): & \\ & x \in_R G \\ & y \in_R G \\ & z = xy \bmod m \\ & X = g^x;\ Y = g^y;\ Z = g^z \\ & d = A(X, Y, Z) \\ & \text{return } d \end{aligned}$$

- Experimento en mundo 0:

$$Exp_{G,g}^{ddh-0}(A):$$
$$x \in_R G$$
$$y \in_R G$$
$$z \in_R G$$
$$X = g^x;\ Y = g^y;\ Z = g^z$$
$$d = A(X, Y, Z)$$
$$\text{return } d$$

La ventaja del adversario es la función definida por:

$$Adv_{G,g}^{ddh} = |Pr[Exp_{G,g}^{ddh-1}(A) = 1] - Pr[Exp_{G,g}^{ddh-0}(A) = 1]|$$

Si la ventaja es baja para todo adversario con recursos (tiempo de ejecución, memoria) razonables se dice que el problema DDH es difícil en G.

2.2. Encriptación

Para entender los protocolos de encriptación utilizados en este trabajo es necesario entender en qué consiste un esquema de encriptación, en particular un esquema de encriptación asimétrica.

Definición: Un esquema de enciptación asimétrica o esquema de clave pública $AE = (K, E, D)$ consiste en tres algoritmos:

1. Un algoritmo de generación de claves aleatorio K, que no toma entradas y retorna un par de claves (PK, SK), la clave pública y la clave privada.
2. Un algoritmo aleatorizado de encriptación E que toma como entradas el texto plano (mensaje) M y la clave pública PK y genera como salida el texto cifrado C o bien $\perp$.
3. Un algoritmo de desencriptación usualmente determinístico D, que toma como entradas el texto cifrado C y la clave secreta SK y da como salida el texto plano M o bien $\perp$.

2.2.1. ElGamal

En 1985 se propuso el sistema de encriptación ElGamal [5], basado en el protocolo de acuerdo de claves públicas de Diffie Hellman [19]. El escenario es el siguiente: Se tiene a Bob el cual desea enviar un mensaje M a Alicia.

Inicialmente, se fijan p, q primos suficientemente grandes tales que q divide a $p-1$, g un generador de un subgrupo G de Z_p^* de orden q. Además, se escoge un valor x_a perteneciente a Z_q aleatoriamente. El valor x_a corresponde a la clave privada sk de Alicia. Su clave pública pk es:

$$Y_a = g^{x_a} \bmod p$$

Luego, Bob envía a Alicia el mensaje M encriptado utilizando el siguiente algoritmo:

$$C = (C_1, C_2) = E_{pk}(M) = (g^r, Y_a^r \cdot M) \bmod p$$

C es el mensaje M encriptado utilizando ElGamal. El mensaje M debe pertenecer a G y r debe pertenecer a Z_q. Alicia, al recibir C utiliza su clave privada y el algoritmo D para obtener M.

$$\begin{aligned} D_{sk}(C_1, C_2) &= C_2 \cdot C_1^{-x_a} \bmod p \\ &= (Y_a^r \cdot M) \cdot (g^r)^{-x_a} \bmod p \\ &= Y_a^r \cdot M \cdot g^{-r x_a} \bmod p \\ &= Y_a^r \cdot M \cdot Y_a^{-r} \bmod p \\ &= M \bmod p \end{aligned}$$

La dificultad de quebrar ElGamal (vulnerar su privacidad) se basa en el supuesto DDH sobre el subgrupo G. ElGamal tiene dos propiedades interesantes, éstas son:

- Reencriptación Pública: Un personaje C que sólo conozca la clave pública puede reencriptar un texto cifrado sin necesidad de conocer la clave privada.

  $$\begin{aligned} (C_1, C_2) &= (g^r, Y_a^r \cdot M) \\ (C_1', C_2') &= ((g^r) \cdot (g^s), Y_a^r \cdot Y_a^s \cdot M) \\ &= (g^{r+s}, Y_a^{r+s} \cdot M) \\ &= (g^{r'}, Y_a^{r'} \cdot M) \end{aligned}$$

  Por lo tanto, (C_1, C_2) y (C_1', C_2') son dos formas distintas de encriptar el mismo texto plano M, y bajo DDH es muy difícil determinar que lo son.

- Homomorfismo: Si (C_1, C_2) y (C_3, C_4) representan la encriptación de M_1 y M_2 respectivamente, entonces $(C_1 \cdot C_3, C_2 \cdot C_4)$ representa la encriptación de $M_1 \cdot M_2$. Una propiedad similar se cumple también para la división.

2.2.2. RSA-OAEP

El algoritmo de encriptación asimétrica RSA fue descrito en el año 1977 por Ron Rivest, Adi Shamir y Leonard Adleman. A continuación, se detallan los tres algoritmos que lo componen.

1. **Generación de claves y parámetros iniciales:**
   - Se escogen dos primos distintos p y q grandes.
   - Se calcula $N = pq$ llamado módulo RSA.
   - Se calcula $\phi(N) = (p-1)(q-1)$.
   - Se escoge un entero positivo $e \in (1, \phi(N))$ tal que $mcd(e, \phi(N)) = 1$.
   - Se calcula d tal que $d \cdot e \equiv 1 \pmod{\phi(N)}$.
   - Clave Pública $= (N, e)$.
   - Clave Privada $= (p, q, d)$.
2. **Encriptación de mensajes:**
   - Recibe como entrada la clave pública (N, e) y un mensaje M, $0 \leq M < N$.
   - Se calcula $C = M^e \ mod \ N$.
   - La salida es el texto cifrado C.
3. **Desencriptación de mensajes:**
   - Se recibe como entrada el módulo público N, la clave privada d, y el texto cifrado C, $0 \leq C < N$.
   - Se calcula $M = C^d \bmod N$.
   - Se entrega M como salida.

RSA por sí solo utiliza un algoritmo de encriptación determinista. Para transformar este algoritmo determinista en un algoritmo probabilista y, además, estandarizar el largo de los mensajes que serán encriptados se utiliza un esquema de *padding* denominado OAEP. El detalle de este esquema se puede apreciar en la figura 2.1.

Donde M es el mensaje sobre el que se desea hacer *padding*, $MGF1$ y $MGF2$ son funciones fijas con descripción pública, *seed* es escogido al azar y PS es la representación en ocho bits del valor hexadecimal 0x01.

En un esquema de encriptación RSA-OAEP se combinan los dos esquemas mencionados anteriormente. Inicialmente se obtiene un *padding* del mensaje M por medio de el esquema de *padding* OAEP obteniendo el valor EM. Este valor es luego encriptado utilizando el algoritmo de encriptación RSA que retornará un valor C. Este valor corresponde al cifrado de M utilizando el esquema de encriptación RSA-OAEP.

2.2.3. Esquema de encriptación DGS

Esquema propuesto en [15]. Es una variación de ElGamal y Paillier [6].

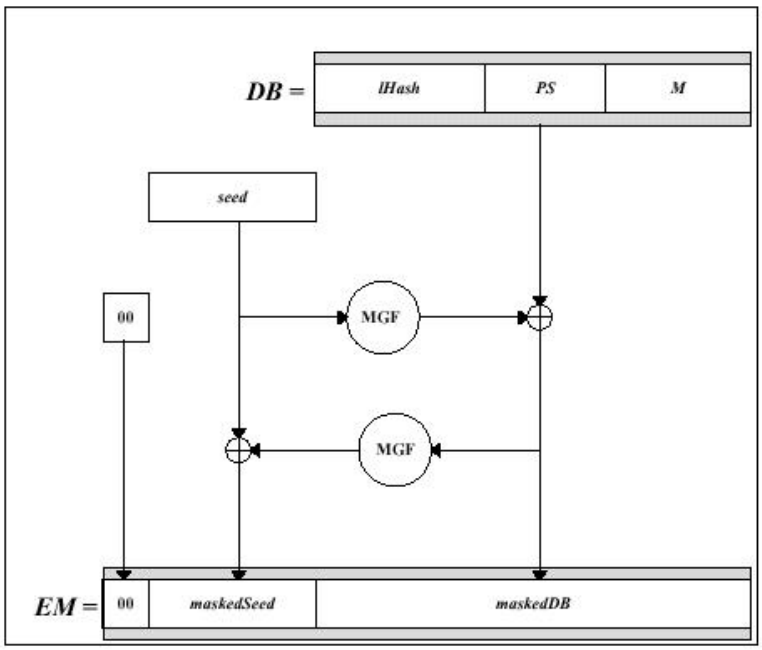

Figura 2.1: Esquema de *padding* OAEP.

Sea $R = Z^*_{N^{s+1}}$ un anillo, donde $N = pq$ es módulo RSA[1] y $s \geq 1$. Se escoge $g \in R$, tal que g tenga orden máximo sobre R. Es decir, cuyo orden sea $\theta = N^s(p-1)(q-1)/2$. Sea $G =< g >$ grupo cíclico, y un número $T \geq \theta^2$. Se puede definir un esquema de encriptación similar al presentado en ElGamal donde la clave pública pk es (R, g, $h = g^x$) donde x es escogido de manera aleatoria en $[0..T]$. La clave privada es x y será distribuida entre las autoridades (ver sección 2.6).

Para encriptar, se debe escoger un r al azar $\in [0..T]$ y un valor $u \in G$. La encriptación se define como $E_{pk}(m, r) = (g^r \bmod N^{s+1}, u^m h^r \bmod N^{s+1})$ donde el espacio de los mensajes corresponde a $Z_{ord(u)}$. Esta variación de ElGamal permite aplicar la propiedad homomórfica a la adición, permitiendo sumar los votos sin necesidad de desencriptarlos. Es decir, $E(m_1, r_1)E(m_2, r_2) = E((m_1 + m_2) \bmod ord(u), r_1 + r_2)$.

La desencriptación se llevará a cabo de manera distribuida por dos autoridades, esto se explicará más adelante. Por ahora se supondrá que ésta se lleva a cabo por una sola entidad.

La desencriptación de un mensaje utilizando lo estudiado en ElGamal (sección 2.2.1) entrega el valor de u^m. Ésto es un trabajo difícil de resolver ya que desencadenaría en el problema del logaritmo discreto. Sin embargo, en algunos anillos existen elementos para los cuales computar el logaritmo discreto es fácil. Para ésto, se escoge $u = \alpha + \beta$. Utilizando la expansión binomial estándar se obtiene que:

$$u^i = (\alpha + \beta)^i = \sum_{j=0}^{i} \binom{i}{j} \alpha^j \beta^{i-j}$$

[1] N corresponde a la multiplicación de dos primos seguros, es decir, primos p y q con mcd($p-1$,$q-1$)=2.

Si $\alpha^j = 0$ para algún j pequeño entonces este valor es sencillo de calcular ya que muchos términos en la expansión desaparecen haciendo este cálculo eficiente, ver sección 3.3.8.

Representación del voto

La representación de la preferencia de un votante se hace mediante un entero que sigue un formato previamente definido. Esta representación debe seguir un formato compatible con la propiedad homomórfica de la encriptación. Sea M una cota superior en el número de votantes, y L el número de candidatos, que debe incluir junto a los candidatos valores adicionales como la opción de votar nulo o blanco si corresponde. Sea j la representación de un candidato $j \in [0..L-1]$ entonces se puede definir un voto para el candidato j como M^j. Luego, si un votante decide votar por el subconjunto de candidatos $j_1..j_\rho$ (donde ρ es la cantidad de candidatos por los cuales se permite votar) un voto o preferencia de un votante se representa como la suma de las preferencias individuales para cada uno de los ρ candidatos escogidos. Es decir, un voto es de la forma $M^{j_1} + ... + M^{j_{rho}}$. El resultado final de las elecciones tendrá la forma $v_0M^0 + v_1M^1 + ... + v_{L-1}M^{L-1}$. Donde v_j es el número de votos para el candidato j.

2.3. Firmas digitales

A continuación se definirá formalmente un esquema de firma digital:

Definición: Un esquema de firma digital $DS = (K, Sign, VF)$ consiste en tres algoritmos:

1. Un algoritmo de generación de claves aleatorio K que no toma entradas y retorna un par de claves (PK, SK), la clave pública y la clave privada.
2. Un algoritmo de firma $Sign$ que toma como entradas el mensaje a firmar M y la clave privada SK y genera como salida la firma tag. El algoritmo, además, puede ser aleatorizado o guardar estado.
3. Un algoritmo de verificación determinístico VF, que toma como entradas el mensaje M, la firma digital tag y la clave pública PK y da como salida la aceptación o rechazo de la firma.

2.3.1. RSA-PSS

El esquema de firmas digitales RSA está compuesto por tres algoritmos. Éstos son:

1. **Generación de claves y parámetros iniciales:**

- Se escogen dos primos distintos p y q grandes.
- Se calcula $N = pq$ llamado módulo RSA.
- Se calcula $\phi(N) = (p-1)(q-1)$.
- Se escoge un entero positivo $e \in (1, \phi(N))$ tal que $mcd(e, \phi(N)) = 1$.
- Se calcula d tal que $d \cdot e \equiv 1 \pmod{\phi(N)}$.
- Clave Pública = (N, e).
- Clave Privada = (p, q, d).

2. **Generación de la Firma Digital:**
   - Recibe como entrada el módulo público N, el valor privado d y el mensaje M que se desea firmar, $0 \leq M < N$.
   - Se calcula $S = M^d \bmod N$.
   - La salida es el par (M, S).

3. **Algoritmo de Verificación:**
   - Se recibe como entrada el módulo público N, la clave pública e, y el par (M, S), $0 \leq S < N$.
   - Se calcula $M' = S^e \bmod N$.
   - La firma es aceptada si $M = M'$.

Para agregar aleatoriedad a la firma y estandarizar el largo de los mensajes que serán firmados se utiliza un esquema de firmas digitales que incluye un *padding* inicial de mensaje. El esquema de firmas digitales utilizado se denomina RSA-PSS. Éste incluye un esquema de *padding* y se puede apreciar en la figura 2.2.

En el esquema de firmas digitales RSA-PSS se debe hacer inicialmente un *padding* sobre el mensaje inicial M. Éste se realiza siguiendo los pasos en el esquema descrito en la figura 2.2. En éste, *hash* corresponde a una función de *hash*, MGF es una función fija de conocimiento público, *padding*1 y *padding*2 son valores fijos definidos inicialmente y *salt* es un valor escogido al azar. Una vez obtenido el valor EM éste es firmado utilizando el algoritmo de firmas digitales RSA.

2.4. Commitments

Para comprender el uso que se da a un *commitment* en criptografía se puede imaginar a un sujeto, el cual dentro de un conjunto finito de elementos o mensajes debe escoger a uno de ellos. Al hacer un *commitment* del elemento escogido, el sujeto se compromete con él de tal manera, que más adelante no puede cambiar su elección. El sujeto así evita revelar su elección inicialmente, puede hacerlo posteriormente simplemente abriendo el *commitment*. Un ejemplo sencillo de ésto es imaginar que Alicia y Bob se encuentran conversando a distancia

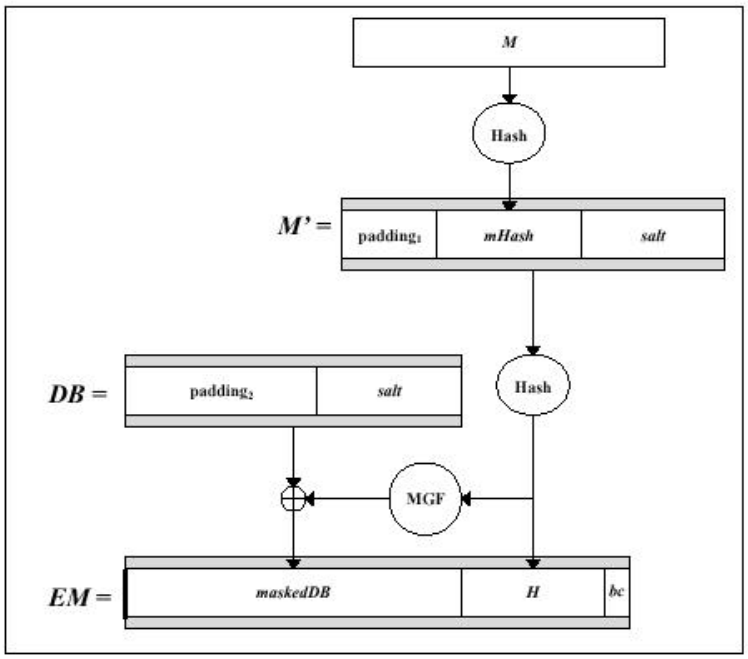

Figura 2.2: Esquema de firmas digitales.

y desean apostar mil dólares a través de un juego. Alicia debe escoger entre dos números. Luego, Bob debe enviar el valor que él cree que Alicia escogió y si logra adivinar gana la apuesta, de lo contrario, gana Alicia. Para ésto, Alicia escoge uno de los dos valores y envía a Bob un *commitment* del escogido. En seguida, Bob envía a Alicia el valor que él cree que ella escogió. Posteriomente, Alicia envía la "llave" para abrir el *commitment* a Bob. Lo anterior funciona ya que un *commitment* satisface dos fuertes propiedades:

- *Binding*: Un *commitment* no puede ser abierto de dos maneras distintas, ésto es, revelando dos valores distintos.
- *Hiding*: No se puede deducir información del contenido de un *commitment* c a partir del valor de éste.

Con respecto a los *commitments* que serán utilizados se define el espacio de los mensajes como M_{com}, un espacio R_{com} de los números que serán escogidos al azar, el espacio de los *commitments* abiertos B_{com} y el espacio de los *commitments* C_{com}. También, se define la función $com(\cdot,\cdot) : M_{com} \times R_{com} \rightarrow C_k$. En este trabajo se considerará $M_{com} = \mathbb{Z}$.

Además, se necesita la propiedad de *root opening* la que especifica que para cualquier $c \in C_{com}$ si es posible encontrar $e \in \mathbb{Z} \setminus \{0\}$ y $m \in \mathbb{Z}$, $z \in B_k$ tal que $com(m; z) = c^e$ entonces se puede computar una forma de abrir c.

El *commitment* a utilizar que cumple con estas propiedades consiste en que dado un N escogido como el producto de dos grandes primos, y dos cuadrados g, h tales que $log_g h$

y $log_h g$ no son conocidos por quien envía y genera el *commitment*. Un *commitment* para un entero m, escogiendo r al azar es $com(m;r) = g^m h^r \mathrm{mod}(N)$.

2.5. Zero-knowledge proofs y protocolos-Σ

Para entender mejor el concepto de *zero-knowledge proof* se debe comprender el concepto de *proof of knowledge*. Una *proof of knowledge* es un protocolo interactivo mediante el cual el *prover* quiere demostrar al *verifier* conocimiento de algo. Sea $L \subseteq \{0,1\}^*$ un lenguaje definido por una relación R ($L = \{x \in \{0,1\}^* / \exists w \in \{0,1\}^* \,\mathrm{tq}\, R(x,w)\}$). Esta relación R es eficientemente computable y "difícil", ésto es, dado un $x \in L$, calcular cualquier w' que satisfaga $(x,w') \in R$ es infactible. El *prover* y el *verifier* conocen un valor x, y el *prover* tiene, además, un testigo o *witness* w tal que $(x,w) \in R$. Una *proof of knowledge* cumple con las siguientes propiedades:

- Completitud: Si el *prover* conoce w tal que $(x,w) \in R$ el *prover* siempre puede convencer al *verifier* que conoce w (lo contrario puede ocurrir con una probabilidad despreciable).
- Validez especial: Ningún *prover* que desconozca w puede convencer al *verifier* que tiene conocimiento de éste (sólo con una probabilidad despreciable).

Una *proof of knowledge* se dice que es *zero-knowledge proof* ante verificador honesto si, sumado a las propiedades anteriores, cumple con la siguiente propiedad:

- Nula divulgación ante verificador honesto: Si el *prover* conoce w y desea demostrarlo al *verifier* puede hacerlo de manera que el *verifier* no pueda aprender nada más que la veracidad de este hecho, ésto es, que existe un w tal que $(x,w) \in R$. Formalmente, equivale a decir que dado $x \in L$, siempre existe un simulador que no necesita el secreto w para generar un grupo de preguntas y respuestas que lucen como una interacción entre el *prover* y el *verifier*. Ésto puede hacerlo con la misma distribución de probabilidad que en las pruebas reales con cualquier w.

Los protocolos-Σ [20] pueden verse como un caso particular de *zero-knowledge proof* que poseen una estructura que consta de tres pasos. Tanto el *prover* como el *verifier* poseen una entrada común x. Los pasos son:

- Commitment inicial: el *prover* envía un *commitment* inicial a.
- Desafío: el verifier responde con un desafío e aleatorio.
- Respuesta: el *prover* responde al desafío con un valor generado utilizando su valor secreto w y el *commitment* inicial a.

Este protocolo puede hacerse no interactivo utilizando la heurística de Fiat Shamir [21, 22]. Bajo esta heurística, se utiliza un desafío $e = h(x, id, a)$, donde id es la identidad del *prover*. Ésto permite hacer a este protocolo no interactivo. La función h es una función de *hash* criptográfica. Se propone además utilizar *commitments* para así hacer el proceso de verificación más fácil. La idea de ésto es utilizar *commitments* que respeten la propiedad homomórfica. En [15] se propuso que dado un *commitment* y una encriptación que contienen el mismo valor, comprobar que el valor que contienen es el mismo y luego trabajar directamente con el *commitment* para así verificar que éste contiene un valor válido. La razón para ésto, es que es significativamente más sencillo trabajar con *commitments* en este proceso, ya que involucran una menor cantidad de exponenciaciones en su cálculo.

2.5.1. Validez de la papeleta de votación

La papeleta de votación es aquello que el votante genera y publica, en ella va incluida su preferencia junto a aquellos valores que garantizan que su voto es correcto. Éstos son:

- Voto Encriptado: El voto encriptado es un par de valores $(g^r, u^m h^r) \in Z^*_{N^{s+1}} \times Z^*_{N^{s+1}}$ según los parámetros definidos en sección 2.2.3. Donde m es la representación del voto y r es un valor escogido de manera aleatoria en [0..T].
- *Proof*: Esta es una ZKP que demuestra que el valor que contiene el voto encriptado es un voto válido para múltiples candidatos. Esta *proof* está formada por *commitments* y un protocolo-Σ que pueden verse como una entrada común conocida tanto por el votante como por las autoridades.
- Firma Digital: Corresponde a una firma digital de los dos valores anteriores (voto encriptado y *proof*) generada por el votante.

Cuando el número de candidatos es grande, los procesos de encriptación se hacen más difíciles de computar, sin embargo, no es en el proceso de encriptación del voto en el que se debe poner mayor atención, sino las *zero-knowledge proof* ya que involucran varias operaciones de aritmética modular y la cantidad de ellas se incrementa si aumenta el número de candidatos.

Se define el concepto de *shadow*, que será utilizado más adelante. Sea v entero cuyo valor es privado, se dice que sh es *shadow* de v si, al sumar ambos valores, no es posible, salvo con una probabilidad despreciable, deducir información sobre el valor v. Se denotará $sh \xrightarrow{sh} v$ a la relación antes mencionada. Se considera el conjunto de *shadow* de v como $sh \in [-2^{|v|+l_s}, 2^{|v|+l_s}]$, donde ls debe ser de un largo suficiente para impedir que información acerca de v sea filtrada. Se definen, además, los conjuntos: $R_{sh} = \{-2^{|N|+l_e+2l_s}, .., 2^{|N|+l_e+2l_s}\}$, $M_{sh} = \{-2^{|N|+l_e+l_s}, .., 2^{|N|+l_e+l_s}\}$, $R_{E,sh} = \{-2^{l_E+l_e+l_s}, .., 2^{l_E+l_e+l_s}\}$. Donde $l_E = \log_2 T$, $l_s = 80$, $|N| = k$ (k parámetro de seguridad).

A continuación, se detallarán las *zero-knowledge proof* utilizadas para el criptosistema presentado anteriormente.

En este caso mediante las *zero-knowledge proof* el votante debe convencer a un verificador (el servidor de escrutinio) que su encriptación posee un voto válido. Ésto se hace mediante un protocolo-Σ. Con los tres valores generados a partir del protocolo, el verificador comprueba si el valor encriptado contiene o no un voto válido.

Se define M como una cota superior al número de votantes y L como el número de opciones o candidatos posibles. Se habla de opciones ya que en ella se debe considerar aquellas decisiones adicionales como votar nulo o blanco dependiendo de las reglas del proceso electoral.

En una *zero-knowledge proof* (ZKP) el *prover* (el votante) quiere convencer a un *verifier* (las autoridades) que "algo" es verdad. En este caso se quiere probar que aquel valor que se envía encriptado contiene un voto válido. En otras palabras, que el valor encriptado contiene un voto de la forma $M^{j_1} + .. + M^{j_\rho}$, donde $\rho \geq 1$ representa al número de opciones por las cuales un candidato puede votar, ya que hablamos de multicandidatos.

Para implementar las ZKP se utilizan los protocolos-Σ. Este protocolo puede ser o no interactivo. En [15] se propone hacerlo no interactivo siguiendo la heurística de Fiat-Shamir utilizando $h : \{0,1\}* \rightarrow \{0,1\}^{l_e}$, una función de *hash* (por ejemplo, SHA256 y con ello $l_e = 256$). Los protocolos-Σ funcionan tanto con encriptación homomórfica, como con *commitment* homomórficos. Debido a la eficiencia de trabajar con *commitments*, una forma de probar la validez de un voto, es demostrar que un *commitment* y una encriptación poseen el mismo valor y luego trabajar directamente con *commitments*.

Para entender mejor el funcionamiento de las *proof* utilizadas se explicarán distintas *proofs* de manera gradual, partiendo de los casos básicos que se desean demostrar, hasta llegar a la *proof* que será finalmente utilizada, que estará compuesta por las anteriores. La forma de abordar el problema será la siguiente:

- Paso 1: Se detallará una *proof* mediante la que es posible demostrar que un voto encriptado y un *commitment* de un voto contienen el mismo valor.
- Paso 2: Se detallará una *proof* con la que se demostrará que existe una relación multiplicativa entre tres *commitments*, vale decir, que el valor de un tercer *commitment* corresponde a la multiplicación de los valores que contienen los otros dos *commitments*.
- Paso 3: Se construirá una *proof* usando las dos *proof* anteriores para demostrar que un valor encriptado posee un voto válido para un candidato u opción.
- Paso 4: Finalmente, se explicará cómo combinar todo lo anterior, para construir una *proof* que permita demostrar que un valor encriptado contiene un voto válido para múltiples candidatos.

PROVER		**VERIFIER**
Entrada común: C, E		
Entrada privada: $m \in \mathbb{Z}_N$, $r_c \in_R R_{com}$, $r_E \in_R R_E$		
1 - Mensaje inicial: $a_c = com(d; r'_c)$ $a_E = E_{pk}(d \bmod N; r'_E)$	$\xrightarrow{a_c, a_E}$	
$d \in_R M_{sh}$; $r'_c \in_R R_{sh}$; $r'_E \in_R R_{E,sh}$		
	$\xleftarrow{e}$	**2 - Desafío:** $e = h(Id, C, E, a_c, a_E)$
3 - Respuesta: $D = em + d$ $z_c = r'_c + er_c$ $z_E = r'_E + er_E$	$\xrightarrow{D, z_c, z_E}$	
		4 - Acepta si: **1**. $(D, z_c, z_e) \in \mathbb{Z} \times R_{com} \times R_E$ **2**. $com(D, z_c) = a_c C^e$ **3**. $E_{pk}(D \bmod N; z_E) = a_E E^e$

Cuadro 2.1: Prueba de un *commitment* y una encriptación conteniendo el mismo elemento mod(N).

Prueba de un *commitment* y una encriptación conteniendo el mismo elemento mod(N):

En esta sección se explicará el primer paso, en el que se demuestra que, dado un *commitment* generado con *com* y un texto encriptado usando el esquema de la sección 2.2.3, ambos contienen el mismo elemento.

Observando el cuadro 2.1, se puede ver el protocolo-Σ utilizado en detalle para esta *proof*. Inicialmente se tienen entradas en común para el *prover* y el *verifier*, que consisten en el voto encriptado $E = E_{pk}(m; r_E)$ y el *commitment* del voto $C = com(m; r_c)$. Para ambos se escogen valores aleatorios (r_c, r_E) que sólo el *prover* conoce. En esta *proof* $d \xrightarrow{sh} em$, $r'_c \xrightarrow{sh} er_c$, $r'_E \xrightarrow{sh} er_E$.

El desafío e debe corresponder a un *hash* (de acuerdo a la heurística de Fiat-Shamir) de los valores Id, C, E, a_c, a_E donde Id es el identificador del prover (votante). Este valor no puede ser calculado de manera maliciosa y se debe comportar como un aleatorio en el oráculo aleatorio [21].

Los valores D, Z_c, Z_E sólo pueden ser generados de manera correcta si el *prover* conoce el valor de m (que debe ser el mismo que contiene C y E) y los aleatorios r_c y r_E. Una demostración de que esta *proof* es ZKP está en [15].

PROVER		VERIFIER
Entrada común: C_a,C_b,C_c		
Entrada privada: $a, b \in \mathbb{Z}_N$; $r_a, r_b, r_c \in_R R_{com}$		
1 - Mensaje inicial: $C_d = com(d; r_d)$ $C_{db} = com(db; r_{db})$	$\xrightarrow{C_d, C_{db}}$	
$d \in_R M_{sh}$; $r_d, r_{db} \in_R R_{sh}$		
	$\xleftarrow{e}$	**2 - Desafío:** $e = h(Id, C_a, C_b, C_c, C_d, C_{db})$
3 - Respuesta: $f = ea + d$ $z_1 = r_d + er_a$ $z_2 = fr_b - er_c - r_{db}$	$\xrightarrow{f, z_1, z_2}$	
		4 - Acepta si: **1**. $f \in \mathbb{Z}; z_1\, z_2 \in R_{com}$ **2**. $com(f, z_1) = C_d C_a^e$ **3**. $C_{db} C_c^e com(0; z_2) = C_b^f$

Cuadro 2.2: Prueba de relación multiplicativa entre tres *commitments*.

Prueba de relación multiplicativa entre tres *commitments*:

Esta *proof* demuestra que dados los *commitments* $C_a = com(a; r_a)$, $C_b = com(b; r_b)$ y $C_c = com(ab; r_c)$, el contenido de C_a multiplicado por el contenido de C_b iguala al contenido de C_c. Para ésto, utiliza la propiedad homomórfica de los *commitments*. En esta *proof* $d \xrightarrow{sh} ea$, $r_d \xrightarrow{sh} er_a$, $r_{db} \xrightarrow{sh} -(ea + d)r_b + er_c$. La razón por la que se detalla esta *proof* es por que más adelante se utilizará para demostrar que dado un voto $v = M^j$, éste es válido ya que divide a M^{L-1}. En rigor, para demostrar la relación entre C_a, C_b y C_c (y que sea *proof of knowledge*) debe considerarse una *proof* en la que se demuestra que el *prover* conoce un *opening* o una forma de abrir c_2 ó c_3 (ver discusión en [15]). Después se verá que esta *proof* en este caso no será necesaria (ver *proof* siguiente). Se puede observar el detalle de esta *proof* en el cuadro 2.2.

Prueba de conocimiento de un voto válido para un candidato:

En esta *proof* se combinan los dos casos anteriores. Por un lado, se debe demostrar que el *commitment* y la encriptación del voto contienen el mismo valor. Y por otro lado, demostrar por medio de operaciones entre *commitments*, que el primer *commitment* contiene un voto válido. Es decir, que el valor que contiene el *commitment* divide a M^{L-1} y, además, contiene un voto positivo. Para ésto, se toma $M = p^2$ donde p es un número primo. Ésto se puede hacer ya que M está definido como una cota superior al número de votantes. El

PROVER		**VERIFIER**
Entrada común: E, C_a, C_b		
Entrada privada: $M^j \in \mathbb{Z}_N$; $r_E \in_R R_E$; $r_a, r_b \in_R R_{com}$		
1 - Mensaje inicial: $C_d = com(d; r_d)$ $C_{db} = com(dp^{L-j-1}; r_{db})$ $C_\gamma = com(dp^j + \gamma; r_\gamma)$ $E_\gamma = E_{pk}(dp^j + \gamma \bmod N; r'_\gamma)$	$\xrightarrow[C_\gamma, E_\gamma]{C_d, C_{db}}$	
$d, \gamma \in_R M_{sh}$; $r_d, r_{db}, r_\gamma \in_R R_{sh}$; $r'_\gamma \in_R R_{E,sh}$		
	$\xleftarrow{e}$	**2 - Desafío:** $e = h(Id, E, C_a, C_b, C_d, C_{db}, C_y, E_\gamma)$
3 - Respuesta: $f = ep^j + d$ $z_1 = r_d + er_a$ $z_2 = fr_b - r_{db}$ $z_3 = fr_a + r_\gamma$ $z_4 = er_E + r'_\gamma$ $D = e \cdot M^j + d \cdot p^j - \gamma$	$\xrightarrow[z_3, z_4, D]{f, z_1, z_2}$	
		4 - Acepta si: **1**. $C_d, C_{db}, C_\gamma \in C_{com}$; $E_\gamma \in C_E$ **2**. $f, D \in \mathbb{Z}$; $z_1, z_2, z_3 \in R_{com}$ $z_4 \in R_E$ **3**. $com(f, z_1) = C_d C_a^e$ **4**. $C_{db} com(p^{L-1}; 0)^e com(0; z_2) = C_b^f$ **5**. $com(D; z_3) = C_a^f C_\gamma$ **6**. $E_{pk}(D; z_4) = E^e E_\gamma$

Cuadro 2.3: Prueba de conocimiento de un voto válido para un candidato.

commitment C_c definido en la sección anterior es reemplazado por $com(p^{L-1}, 0)$, es por esta razón que ya no es necesario realizar la *proof* mencionada anteriormente (donde el *prover* demuestra que conoce un *opening* a C_b o C_c), ya que se tiene lo pedido.

Por una parte, se demuestra que el contenido de un *commitment* al cuadrado iguala el voto que se encuentra encriptado y, por lo tanto, es positivo. Por otra, que el voto es de la forma M^j, por lo tanto, es un voto válido.

Observar que hasta acá no se han considerado los multicandidatos. Se definen los valores $E = E_{pk}(M^j; r_E)$, $C_a = com(p^j; r_a)$, $C_b = com(p^{L-j-1}b; r_b)$. Se puede apreciar la *proof* en detalle en el cuadro 2.3. En ella, $d \xrightarrow{sh} p^j$, $\gamma \xrightarrow{sh} eM^j + dp^j$,$r_d \xrightarrow{sh} er_a$,$r_{db} \xrightarrow{sh} (ep^j + d)r_b$,$r_\gamma \xrightarrow{sh} (ep^j + d)r_a$, $r'_\gamma \xrightarrow{sh} er_E$.

Prueba de conocimiento de una encriptación conteniendo un voto válido para múltiples candidatos:

En esta prueba se aplica lo estudiado anteriormente para cada segmento del voto. Ahora se considera un nuevo valor ρ que será el número de candidatos que el votante puede escoger. Estos votos deben realizarse para distintos candidatos. Por lo tanto, el votante debe entregar la suma de la representación del voto de cada uno de sus candidatos escogidos. Es necesario probar que esta suma es correcta. Para demostrarlo se deben generar *commitments* $C_1, ..., C_\rho$ para $p^{j_1}, ..., p^{j_\rho}$. Además, se deben generar *commitments* $C'_1, ..., C'_\rho$ para $p^{j_2-j_1-1}, ..., p^{L-1-j_\rho-1}$. Usando pruebas de multiplicación se puede demostrar que la multiplicación del contenido de C_i y $(C'_i)^p$ es equivalente al contenido de C_{i+1} donde $C_{\rho+1} = E_{pk}(p^L; 0)$. Con lo anterior se demuestra que todos los *comitments* $C_1, ..., C_\rho$ poseen potencias de p y que todos sus exponentes son diferentes y pertenecen al intervalo $[0, L-1]$. Luego, se pueden generar *commitments* $C''_1, ..., C''_\rho$ para $M^{j_1}, ..., M^{j_\rho}$. Con estos valores se puede demostrar para $i = 1..\rho$ que el contenido de $c''_1, ..., c''_\rho$ contiene el cuadrado del contenido de $c_1, ..., c_\rho$. Para demostrar posteriormente, que el voto es de la forma $M^{j_1} + .. + M^{j_\rho}$.

En el cuadro 2.4 se detalla esta *proof*. En ella $E = E_{pk}(M^{j_1} + .. + M^{j_\rho}; r_E)$, $C_i = com(p^{j_i}; r_i)$, $C'_i = com(p^{j_{i+1}-j_i-1}; r'_i)$; $\forall i = 1..\rho$.

En resumen, la papeleta de votación está compuesta por los siguientes valores:

- Voto Encriptado: El voto encriptado corresponde a un par de valores $(g^r, u^m h^r) \in Z^*_{N^{s+1}} \times Z^*_{N^{s+1}}$ según los parámetros definidos en sección 2.2.3. Donde m es la representación del voto y r es un valor escogido de manera aleatoria en [0..T].
- *Proof*: Este valor se utiliza para demostrar que el voto encriptado contiene un voto válido según la representación de un voto definida en la sección 2.2.3. Esta *proof* está compuesta por un mensaje inicial, y una respuesta al desafío generado de manera no interactiva. Esta *proof* es generada de acuerdo al algoritmo descrito en el cuadro 2.4.
- Firma Digital: El votante debe anexar además una firma digital de los dos valores anteriores, para poder verificar que es él quien generó aquellos valores y que sólo haya votado una vez.

2.6. Protocolo DKG

Este protocolo permite generar una clave distribuida para criptosistemas basados en logaritmo discreto. La idea de ésto, es generar una clave x distribuida entre n autoridades tal que ninguna por sí sola pueda obtener ninguna información acerca de esta clave pero sí puedan recontruirla un subconjunto $t+1$ de ellas con $n/2 - 1 \leq t \leq n$. El sistema, por lo tanto, permite que una cantidad t de autoridades puedan estar comprometidas. Se basa en lo estudiado en [23] y [24]. Inicialmente se consideró implementar lo estudiado en [23], sin embargo, presenta algunas vulnerabilidades a nivel de seguridad que permiten a un

PROVER		VERIFIER
Entrada común: E,C_i,C'_i $(\forall i = 1..\rho)$		
Entrada privada: $M^{j_i} \in \mathbb{Z}_N$; $r_E \in_R R_E$; $r_i, r'_i \in_R R_{com}$ $(\forall i = 1..\rho)$		
1 - Mensaje inicial: $C_{di} = com(d_i; r_{di})$ $C_{dib} = com(d_i p^{j_{i+1}-j_i}; r_{dib})$ $C_\gamma = com(\gamma; r_\gamma)$ $E_\gamma = E_{pk}(d_1 p^{j_1} + .. + d_\rho p^{j_\rho} + \gamma \bmod N; r'_\gamma)$ $d_i, \gamma \in_R M_{sh}$; $r_{di}, r_{dib}, r_\gamma \in_R R_{sh}$; $r'_\gamma \in_R R_{E,sh}$ $(\forall i = 1..\rho)$	$\xrightarrow[C_\gamma, E_\gamma]{C_{di}, C_{dib}}$	
	$\xleftarrow{e}$	**2 - Desafío:** $e = h(Id, E, C_i, C'_i, C_{di}, C_{dib}, C_\gamma, E_\gamma)$
3 - Respuesta: $f_i = ep^{j_i} + d_i$ $z_{1,i} = r_{d_i} + er_i$ $z_{2,i} = pf_i r'_i - er_{i+1} - r_{dib}$ $z_3 = f_1 r_1 + .. + f_\rho r_\rho + r_\gamma$ $z_4 = er_E + r'_\gamma$ $D = e(M^{j_1} + .. + M^{j_\rho}) + d_1 p^{j_1} + .. + d_\rho p^{j_\rho} + \gamma$ $\forall i = 1..\rho$	$\xrightarrow[z_3, z_4, D]{f_i, z_{1,i}, z_{2,i}}$	
		4 - Acepta si: **1**. $C_{di}, C_{dib}, C_\gamma \in C_{com}$; $E_\gamma \in C_E$ **2**. $f_i, D \in \mathbb{Z}$; $z_{1,i}, z_{2,i}, z_3 \in R_{com}$ **3**. $z_4 \in R_E$ **4**. $com(f_i, z_{1,i}) = C_{di} C_i^e$ **5**. $C_{dib} C_{i+1}^e com(0; z_{2,i}) = C_i'^{pf_i}$ **6**. $com(D; z_3) = C_1^{f_1} .. C_\rho^{f_\rho} C_\gamma$ **7**. $E_{pk}(D; z_4) = E^e E_\gamma$ $\forall i = 1..\rho$

Cuadro 2.4: Prueba de conocimiento de un voto válido para múltiples candidatos.

eventual adversario manipular la salida pública del protocolo de manera que ésta no siga una distribución uniforme sobre el espacio posible de claves.

Sea n el número de autoridades, $k \in \mathbb{N}$ el parámetro de seguridad, N módulo RSA de k bits, $s \geq 1$ entero, $g \in R = Z^*_{N^{s+1}}$ de orden máximo θ (ver sección 2.2.3), $h \in < g >$ tal que $\log_g h$ y $\log_h g$ no son conocidos, t el número de autoridades necesarias para la desencriptación y $T \geq \text{ord}^2(g)$, una cota superior conocida del cuadrado del orden de g. Se detalla el protocolo de generación de claves a continuación:

Participantes Autoridades $P_1, ..., P_n$.
Input públicos n, N, s, g, h, T, t.
Input privados No hay.
Generación de claves

1. - Cada autoridad P_i genera al azar polinomios $f_i(z)$, $f'_i(z)$ en Z_q de grado t tales que:

$$\begin{aligned} f_i(z) &= a_{i0} + a_{i1}z + a_{i2}z^2 + ... + a_{it}z^t \bmod T \\ f'_i(z) &= b_{i0} + b_{i1}z + b_{i2}z^2 + ... + b_{it}z^t \bmod T \end{aligned}$$

   Luego publica $C_{ik} = g^{a_{ik}} h^{b_{ik}} \bmod N^{s+1}$, $\forall k = 0, ..., t$ y envía en privado (utilizando encriptación) a P_j los valores $s_{ij} = f_i(j) \bmod T$, $s'_{ij} = f'_i(j) \bmod T$, $\forall j = 1, ..., n$.
   - Cada autoridad P_j verifica los valores recibidos por las distintas autoridades P_i de la siguiente manera:

$$g^{s_{ij}} h^{s'_{ij}} = \prod_{k=0}^{t} (C_{ik})^{j^k} \bmod N^{s+1} \tag{2.1}$$

   Si la ecuación 2.1 falla para cierto i la autoridad P_j publica una queja contra P_i.
   - Una vez completado el paso anterior, toda autoridad P_i que haya recibido queja de otra autoridad P_j publica los valores s_{ij}, s'_{ij}.
   - Se considera deshonesta a una autoridad si ha recibido más de t quejas en los pasos anteriores o si los valores publicados en el paso anterior no cumplen con la ecuación 2.1.

2. Se define *Qual* como el conjunto de autoridades honestas, ésto se puede calcular públicamente.

3. La clave privada es $x = \sum_{i \in Qual} a_{i0} \bmod \theta$ pero no se calcula directamente. Cada autoridad P_i calcula su *share* como $x_i = \sum_{j \in Qual} s_{ji} \bmod T$.

4. Cada autoridad calificada como honesta en los pasos anteriores publica $A_{ik} = g^{a_{ik}} \bmod N^{s+1}$, $\forall k = 0, ..., t$.

5. Cada autoridad P_j verifica los valores publicados en el paso anterior por las distintas autoridades mediante la siguiente ecuación:

$$g^{s_{ik}} = \prod_{k=0}^{t} (A_{ik})^{j^k} \bmod N^{s+1} \tag{2.2}$$

Si estos cálculos fallan para un cierto i, P_j se queja contra la autoridad P_i y publica los valores s_{ij}, s'_{ij} que satisfacen la ecuación 2.1 pero no satisfacen la ecuación 2.2.

6. Sea *Queja* el conjunto de todas aquellas autoridades P_i. Para aquellas autoridades $P_i \in Qual$ que hayan recibido al menos una queja válida, el resto de las autoridades pertenecientes a $Qual \backslash Queja$ recalculan x_i (y lo hacen público) por medio de la reconstrucción del polinomio $f_i(z)$ utilizando interpolación de Lagrange y los valores $f_i(j)$ que cada autoridad $j \in Qual$ recibió de P_i. Sea $Qual(t+1)$ un subconjunto de $t+1$ autoridades en $Qual$, la fórmula para ésto es:

$$f_i(z) = \sum_{j \in Qual(t+1)} \left(f_i(j) \prod_{k \in Qual(t+1), k \neq j} \frac{z-k}{j-k} \right) \bmod T \tag{2.3}$$

   Con ello, $s_{i,j} = f_i(j) \bmod T$, $x_i = \sum_{j \in Qual} s_{ji} \bmod T$.

7. La clave pública se calcula como $y = \prod_{i \in Qual} A_{i0} \bmod N^{s+1}$.

Output Públicos y, $Qual$, A_{ij} $(\forall i, j = 1, .., n)$, $Queja$, x_k $(k \in Queja)$.
Output Privados s_{ij}, x_i, $\forall i \in Qual \backslash Queja$.

2.7. Protocolo BGW

Este protocolo permite el cálculo de un entero compuesto N de manera distribuida. Se tienen n participantes, y cada uno de ellos posee un par de valores (p_i, q_i) los cuales son de conocimiento único del participante i. Los participantes desean calcular en conjunto el valor:

$$N = (\sum_{i=1}^{n}(p_i))(\sum_{i=1}^{n}(q_i))$$

De manera que al final de este cálculo cada uno de los participantes P_i sólo conozca el valor de N y de su *share* (p_i, q_i). Para ésto, se siguen cuatro pasos. Éstos son:

- **Paso 1**

  Sea $l = \lfloor \frac{n-1}{2} \rfloor$. Cada participante P_i debe escoger $4l$ coeficientes aleatorios $a_{i,1}, ..., a_{i,l}$, $b_{i,1}, ..., b_{i,l}$ y $c_{i,1}, ..., c_{i,2l}$. A continuación, cada participante genera los siguientes polinomios:

$$
\begin{aligned}
f_i(x) &= p_i + \sum_{j=1}^{l}(a_{i,j}x^j) \\
g_i(x) &= q_i + \sum_{j=1}^{l}(b_{i,j}x^j) \\
h_i(x) &= \sum_{j=1}^{2l}(c_{i,j}x^j)
\end{aligned}
$$

Todo participante P_i calcula $\forall j = 1, ..., n$ los valores $f_i(j), g_i(j), h_i(j)$.

- **Paso 2**

  Luego, P_i envía de manera privada los valores $(f_i(j), g_i(j), h_i(j))$ a P_j.

- **Paso 3**

  Una vez que el participante P_j recibe las n tuplas del resto de los participantes calcula:

$$N_j = (\sum_{i=1}^{n} f_i(j))(\sum_{i=1}^{n} g_i(j)) + \sum_{i=1}^{n}(h_i(j))$$

  y publica este valor.

- **Paso 4**

  Sea el polinomio

$$N(x) = F(x)G(x) + H(x)$$

  donde

  - $F(x) = \sum_{i=1}^{n} f_i(x)$
  - $G(x) = \sum_{i=1}^{n} g_i(x)$
  - $H(x) = \sum_{i=1}^{n} h_i(x)$

  Dado el orden de $N(x)$ basta conocer los valores del polinomio N en n puntos para reconstruirlo. Es posible demostrar que:

$$N(0) = (\sum_{i=1}^{n}(p_i))(\sum_{i=1}^{n}(q_i))$$

  Es posible también, que el resultado final sea compartido de manera aditiva entre los n participantes. En otras palabras, que N sea desconocido para cada participante pero que su valor pueda reconstruirse a partir de los *share* n_j de cada uno de ellos. Es decir, $N = \sum_{i=1}^{n} n_j$. Para ésto, los participantes no deben ejecutar el paso 4 ni publicar los valores N_j. En vez de eso, cada autoridad calcula su *share* como:

$$n_j = (\prod_{1 \leq h \leq n, h \neq j} \frac{h}{h-j})N_j$$

2.8. Generación de módulo RSA de manera distribuida

La generación de este módulo se basa en lo estudiado en [25]. Este esquema permite generar un módulo RSA N que corresponde a la multiplicación de dos valores p y q que no tienen factores primos pequeños. El valor N es generado de manera distribuida entre n participantes, por lo tanto, ninguno de los ellos conoce su factorización. Este esquema consiste en tres pasos:

- **Paso 1**
  En este primer paso cada participantes P_i escoge dos valores aleatorios p_i, q_i en el intervalo $[\lfloor 2^{(k-1)/2} \rfloor, \lfloor \frac{2^{k/2}-1}{n} \rfloor]$ (donde k es un parámetro de seguridad) y se verifica que los valores $p = \sum_{i=1}^{n} p_i$ y $q = \sum_{i=1}^{n} q_i$ no tengan factores primos pequeños. Para escoger estos valores, cada participante P_i escoge inicialmente un entero aleatorio a_i en el rango $[1, P]$ que es primo relativo con P, donde P corresponde a la multiplicación de todos los primos impares menores a un valor B definido inicialmente. El producto $a = a_1 \times ... \times a_n \bmod P$ es también relativamente primo a P. Luego, los participantes deben transformar este *sharing* multiplicativo en un *sharing* aditivo. Ésto deben hacerlo de manera iterativa, siguiendo los siguientes pasos:
  - En la primera iteración se tienen los siguientes valores:

    $$\begin{cases} u_{1,i} = a_1,\ v_{1,i} = 1 \,\text{para}\, i = 1 \\ u_{1,i} = 0,\ v_{1,i} = 0 \,\text{para}\, i \neq 1 \end{cases}$$

    Donde

    $$\begin{aligned} a_1 &= (a_1 + 0 + ... + 0)(1 + 0 + ... + 0) \bmod P \\ &= (u_{1,1} + ... + u_{1,n})(v_{1,1} + ... + v_{1,n}) \bmod P \end{aligned}$$

    Los participantes utilizan el protocolo definido en la sección 2.7 sobre este valor, para generar un *sharing* aditivo de éste. Al final de esta iteración se obtienen los siguientes *sharing* para a_1:

    $$a_1 = u_{2,1} + ... + u_{2,n}$$

  - Previo a la i-ésima iteración los participantes estarán compartiendo el valor

    $$a_1 \cdots a_{i-1} = u_{i,1} + ... + u_{i,n} \bmod P$$

    Se tiene que:

    $$\begin{cases} v_{i,j} = a_i \,\text{para}\, j = i \\ v_{i,j} = 0 \,\text{para}\, j \neq i \end{cases}$$

    Posteriormente, los participantes generan un *sharing* aditivo del valor:

    $$\begin{aligned} a_1 \cdots a_i &= (u_{i,1} + ... + u_{i,n})(0 + 0 + ... + a_i + 0 + ... + 0) \bmod P \\ &= (u_{i,1} + ... + u_{i,n})(v_{i,1} + ... + v_{i,n}) \bmod P \end{aligned}$$

mediante el algoritmo descrito en la sección 2.7. Se obtiene el siguiente *sharing* aditivo:

$$a_1 \cdots a_i = u_{i+1,1} + ... + u_{i+1,n} \bmod P$$

Después de realizar los n pasos se obtiene un *sharing* para el valor a:

$$a = u_{n+1,1} + ... + u_{n+1,n} \bmod P$$

Una vez obtenidos los valores $u_i = u_{n+1,i}$ cada participante genera un entero aleatorio r_i y calcula el valor $p_i = r_i P + u_i$.

Para los valores p y q se debe verificar finalmente que $\text{MCD}(p-1, P) = 1$ y que $\text{MCD}(p-1, 4P) = 2$. Esta verificación debe hacerse utilizando un protocolo MCD distribuido [25].

- **Paso 2**

  Se utiliza el protocolo descrito en la sección 2.7 para calcular N a partir de los valores u_i generados en el primer paso.

- **Paso 3**

  Finalmente, los participantes realizan un test que extiende a *Fermat Primality test*[2] [26], mediante ésto, verifican que N corresponde al producto de dos primos de largo suficiente. Los pasos del test se describen a continuación.

  Inicialmente, los participantes acuerdan un entero g tal que $\text{MCD}(g, N) = 1$. Luego, cada participante P_i calcula y publica:

$$v_i = \begin{cases} g^{N-p_1-q_1+1} \bmod P \text{ para } i = 1 \\ g^{p_i+q_i} \bmod P \text{ para } i \neq 1 \end{cases}$$

  Finalmente, los participantes verifican que se cumpla la siguiente igualdad:

$$v_1 = \prod_{i=1}^{n} v_i$$

  Si esta igualdad no se cumple se debe comenzar nuevamente todo el protocolo.

[2]El *Fermat primality test* es un algoritmo de verificación probabilista para determinar si un número es primo con alta probabilidad.

Capítulo 3

Descripción del sistema de votación

En esta sección se explicará la solución en detalle, sus distintas componentes y funcionalidades.

3.1. Arquitectura de la solución

La solución tal como muestra la figura está compuesta de cuatro componentes.

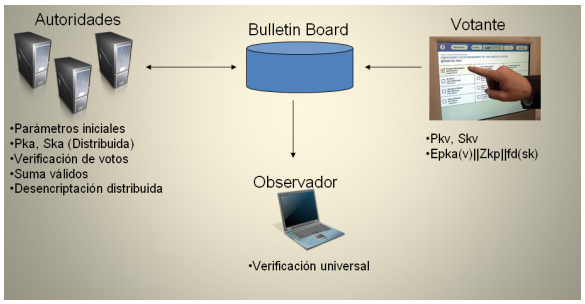

Figura 3.1: Esquema de votación electrónica.

- **BulletinBoard (BB):** Esta componente corresponde a una base de datos pública que permite a cualquier participante leer los valores que hay en ella, pero no modificarlos. En él se publica toda la información necesaria para realizar la verificación universal. El BulletinBoard se implementará como un servidor que recibe valores y los publica. Además, éste se encarga de enviar valores que puedan ser solicitados por las distintas autoridades, observadores y votantes.
- **Software autoridades (SA):** En este esquema se utilizarán tres autoridades. El protocolo debe funcionar correctamente aun si una de ellas se encuentra comprometida. Es decir, el protocolo supone que a lo más una autoridad puede estar comprometida y

que si ésta intenta alterar los resultados no podrá alterar el buen funcionamiento del sistema, sino que sólo será descartada. Las autoridades serán las encargadas de generar los parámetros necesarios para el sistema descritos en la sección 2.2.3, generar el par de claves necesarios para la encriptación y desencriptación de los votos, verificar la validez de las papeletas de votación, sumar y desencriptar el total de votos. Cada una de las acciones que las autoriades llevan a cabo deben ser públicadas en el BB. En caso de tratarse de valores privados, deben emitir *proofs* para garantizar que están actuando correctamente.

- **Software votante (SV):** Este módulo es el encargado de obtener la clave pública generada en el DKG consultándola al BB, generar el voto encriptado reflejando la elección del votante. De igual modo, éste se encarga de generar una ZKP para el voto y una firma digital para verificar que efectivamente el votante fue quien emitió un determinado voto y ZKP. Por último, es el encargado de enviar los datos generados al BulletinBoard para ser publicados.
- **Software observador (SO):** Este *software* permitirá la verificación universal. Es decir, se encarga de verificar la validez de los votos publicados, y de los valores públicos generados por las autoridades en los distintos procesos.

3.2. Descripción general del protocolo

El protocolo a grandes rasgos se puede ver de la siguiente manera:

1. Fase 1: Inicialización
   - Las autoridades mediante SA generan parámetros iniciales que serán usados en el proceso de votación de manera distrubuida, de esta manera ninguna de ellas puede alterar los valores generados para luego manipular los resultados finales. Una vez generados los valores son publicados en el BB.
   - Las autoridades mediante SA generan en conjunto una clave pública y privada de manera distribuida. En cada una de las etapas de este proceso las autoridades deben emitir pruebas de que están actuando honestamente. En este proceso, además, si una autoridad actúa de manera deshonesta es detectado e inmediatamente es descartada. Una vez generados los valores publican la clave pública en el BB. Ver figura 3.2.
2. Fase 2: Generación y depósito de la papeleta de votación
   - El votante por medio de SV genera un voto encriptado con la clave pública generada por las autoridades, ésta la obtiene del BB, genera una ZKP para ese voto encriptado. Éstos dos valores los firma de manera digital con su clave privada. Estos tres valores corresponden a la papeleta de votación y son publicados en el BB. A pesar de que un voto encriptado vaya firmado por el votante, la desencriptación no se llevará a cabo de manera individual, es por esta razón que se conserva la privacidad del voto. Ver figura 3.3.

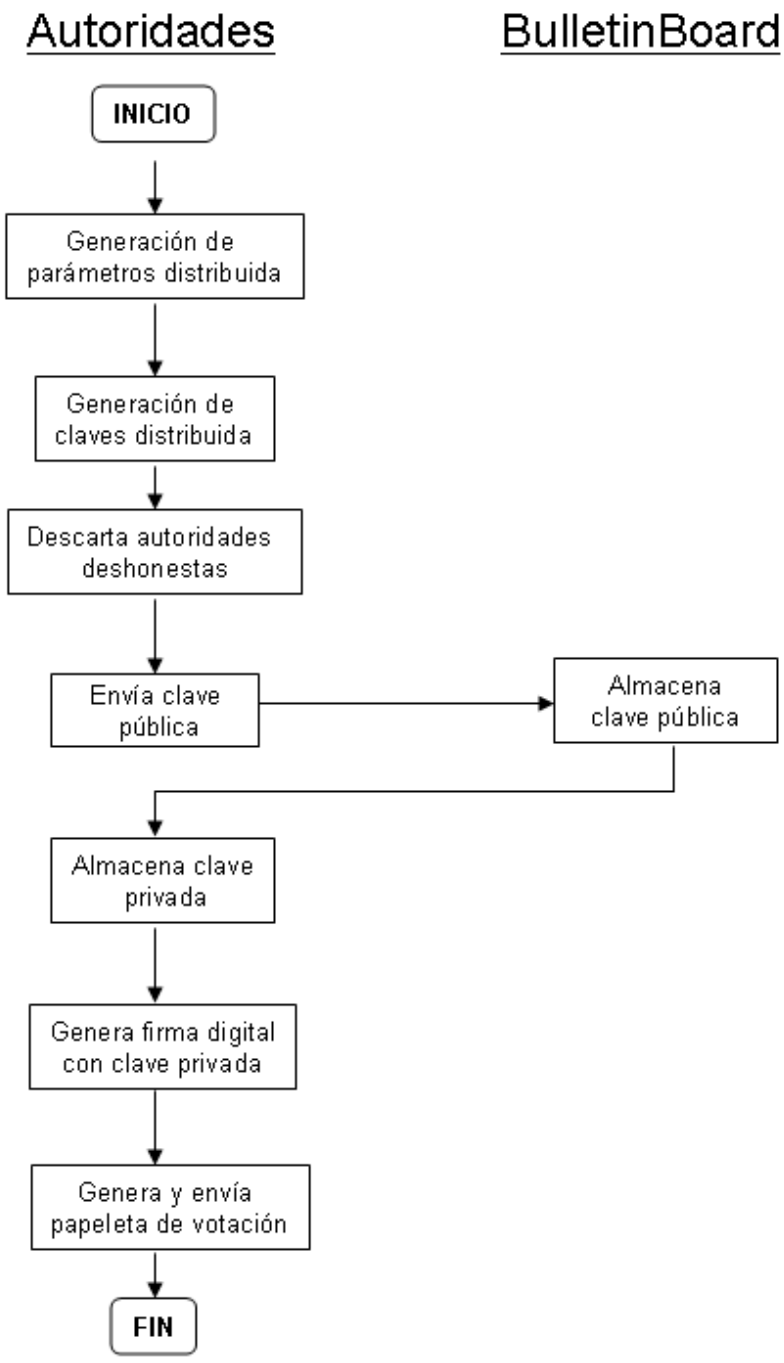

Figura 3.2: Diagrama de flujo: Inicialización

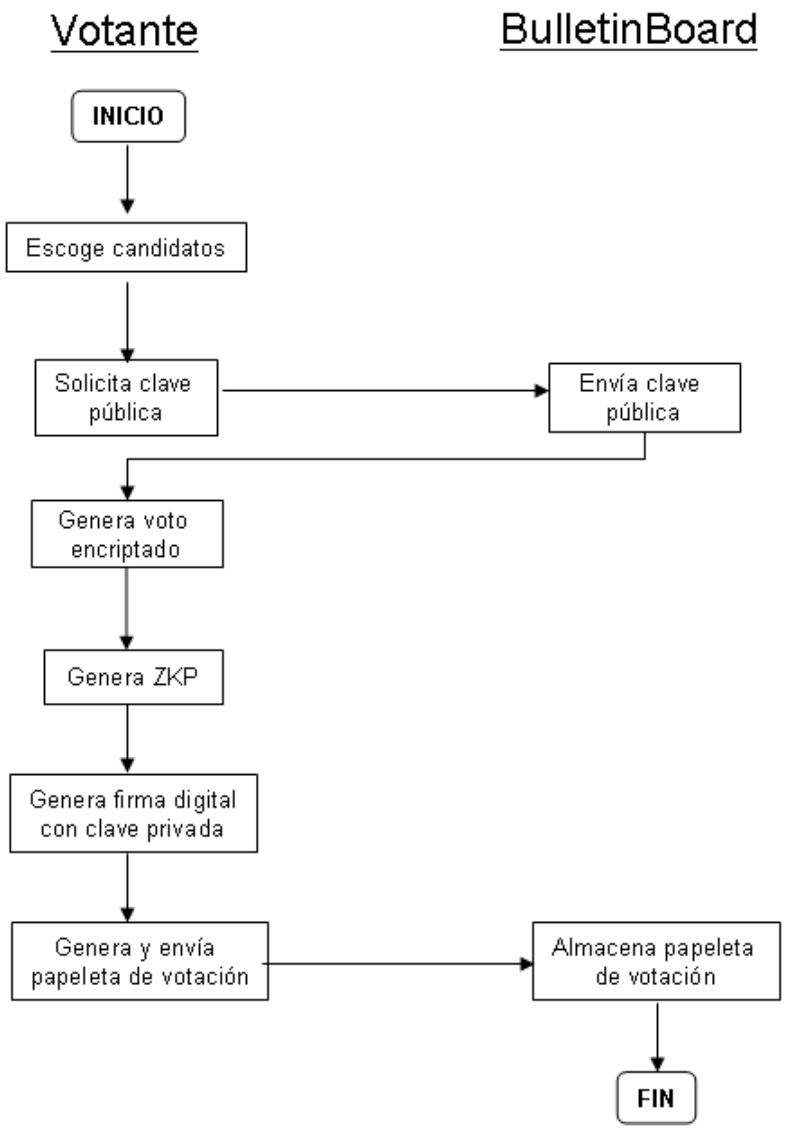

Figura 3.3: Diagrama de flujo: Generación y depósito de la papeleta de votación

3. Fase 3: Verificación de la papeleta de votación

    - Las autoridades mediante SA confirman la validez de los votos verificando que sea considerado sólo un voto por individuo, que la firma corresponda al votante y que la ZKP sea válida. Las Autoridades deben publicar en el BB si un voto es rechazado y el motivo del rechazo. Ver figura 3.4.

4. Fase 4: Escrutinio.

    - Las autoridades calculan la suma encriptada de los votos, dadas las propiedades de la encriptación homomórfica. Son considerados en esta suma aquellas papeletas consideradas válidas por al menos dos autoridades. Ningún voto es desencriptado de manera individual, manteniendo así la privacidad del voto.
    - Dos autoridades son las encargadas de desencriptar el total de votos de manera distribuida (desencriptación por capas), para ello deben además generar una *proof* que garantice que la desencriptación parcial es válida. Ver figura 3.5.

Durante cada una de estas etapas los observadores mediante SO pueden verificar que las operaciones se estén llevando a cabo correctamente. Por ejemplo, que los votos rechazados realmente no correspondan a votos válidos. Gracias a esta propiedad el votante puede convencerse que su voto fue contabilizado correctamente, ya que podrá hacer un seguimiento de su papeleta de votación.

3.3. Protocolo

A continuación se detallarán los procesos involucrados en este esquema de votación electrónica:

3.3.1. Generación de claves de los votantes

Éste es un proceso previo a las elecciones en el que los votantes adquieren un par (clave pública, clave privada). Los valores de éstas (en particular de la clave privada) deben ser almacenadas en un dispositivo seguro, por ejemplo una *smartcard*. De esta manera, dicha clave no puede ser recuperada, sino que sólo es usada para generar firmas (la generación de firmas es realizado dentro de la *smartcard*).

Por simplicidad, en el presente trabajo se supone que la clave privada se almacena en el computador del votante.

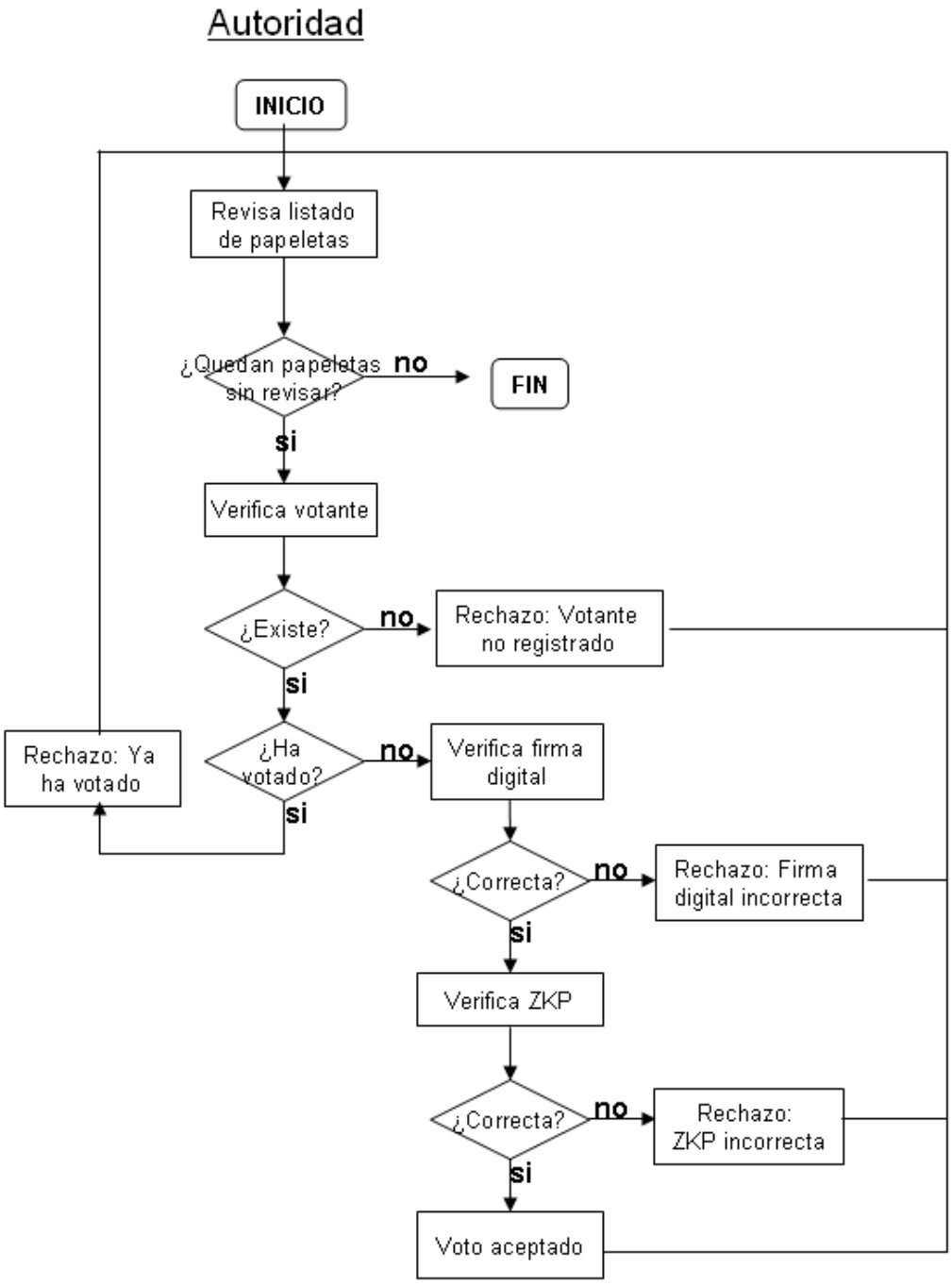

Figura 3.4: Diagrama de flujo: Verificación de la papeleta de votación

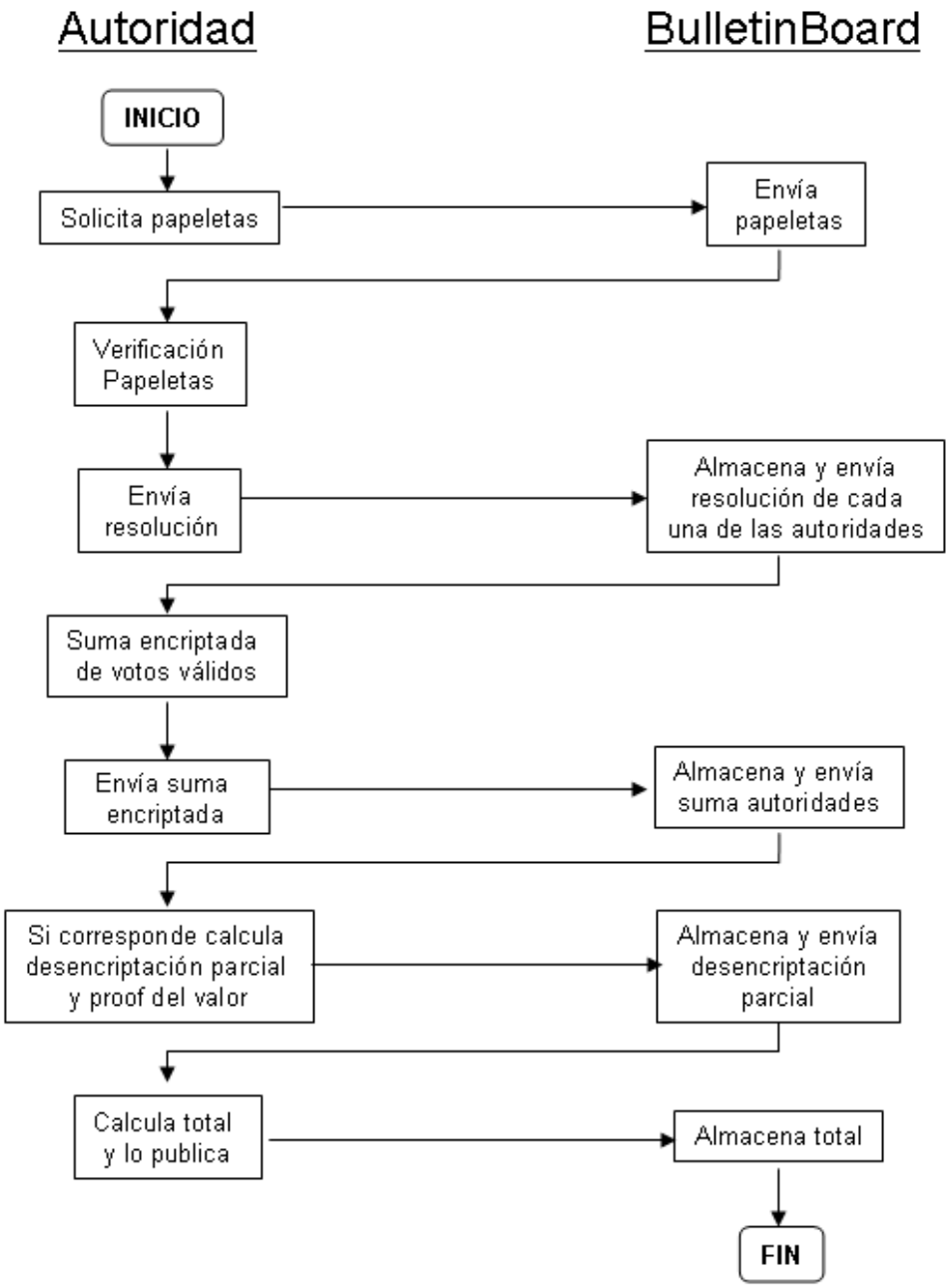

Figura 3.5: Diagrama de flujo: Escrutinio

3.3.2. Generación de parámetros

Esta es la primera etapa en la cual se generan aquellos parámetros que serán utilizados en el resto de la solución. Las autoridades son las encargadas de generar N que corresponde a un módulo RSA, o sea, $N = pq$ con p y q primos. El valor N es generado entre las autoridades de acuerdo al protocolo descrito en la sección 2.8. En este trabajo se considerará el valor de $s = 1$. Con el valor de N y s es posible definir el anillo $R = Z^*_{N^{s+1}}$ sobre el cual se trabajará. Además, se debe escoger un valor $g \in R$ tal que éste sea generador de un subgrupo G de R. Las condiciones de seguridad para g son poseer símbolo de Jacobi igual a 1 y que su orden sea máximo. El orden de R es $N^s(p-1)(q-1)$. Por otro lado, por el teorema de Lagrange se sabe que el orden de un subgrupo debe dividir al orden del grupo. Dado que G es subgrupo de R y que su orden es máximo, el orden de g correponde a $\theta = N^s(p-1)(q-1)/2$. La factorización de N debe ser desconocida (de lo contrario rompería la seguridad del sistema). Es por ésto, que el generador g de G se obtiene mediante la operación $g = (1+N)^e i \bmod N^{s+1}$, con e, i escogidos al azar en $Z^*_{N^{s+1}}$ de acuerdo a lo propuesto en [16]. También, se escoge un parámetro $T \geq \theta^2$. Éste permite que la distribución de elementos de la forma g^a en G sea cercana a uniforme. Por último, se debe generar $h \in$ ¡g¿tal que $\log_g h$ y $\log_h g$ no son conocidos. La elección de estos parámetros se encuentra justificada en [15].

La generación de números aleatorios se debe realizar de manera que cada una de las autoridades participe, evitando así que sólo una de ellas tenga el control de éstos. Para ésto, se sigue el protocolo DKG para producir $y' = g^{x'}$. Luego, se calcula un $hash$ sobre y' generando un *string* σ. El valor σ obtenido debe tener un largo suficiente, ya que cada vez que se necesite un aleatorio las autoridades obtendrán un trozo del valor obtenido del largo deseado.

3.3.3. Protocolo DKG

Para ésto, se utiliza el protocolo DKG detallado en la sección 2.6. Se entrega como entradas $n = 3$, N, $s = 1$, g, h, T, $t = 1$. Se obtiene una clave pública y y el conjunto de *shares* de la clave secreta.

3.3.4. Elección del voto

En esta etapa el votante escoge su voto m. Sea L el número de candidatos u opciones, dentro de las cuales deben considerarse los votos blancos o nulos. Sea M una cota superior al número de votantes. El voto m tiene la forma $M^{j_1} + ... + M^{j_\rho}$ con $0 \leq j_1, ..., j_\rho \leq L-1$, donde ρ es el número de candidatos que se permite escoger y cada j_i corresponde al número asociado a un candidato escogido por el votante. Por ejemplo, si existen 5 candidatos y se permiten los votos blancos y nulos entonces $L = 7$. En el caso en que se permite votar por 3 candidatos entonces $\rho = 3$. Si el votante desea votar por los candidatos 1, 2 y 4 su voto se representa de la forma $M^0 + M^1 + M^3$ ya que cada candidato se representa con su valor menos uno.

Para la encriptación se tienen los valores de $u = N + 1$, $ord(u) = N^s$. Luego, el votante por medio de SV genera un valor aleatorio $r \in [0..T]$. La encriptación se calcula como $E(m, r) = (g^r, u^m g^x) \bmod N^{s+1}$. Se publica este valor junto a la ZKP generada mediante el algoritmo descrito en la sección 2.5.1 y una firma digital del votante en el BB (la papeleta de votación). Para la generación de esta firma digital se utilizará la clave privada del votante generada en la primera etapa.

3.3.5. Revisión de votos válidos

En esta etapa las autoridades verifican las papeletas de votación publicadas en el BB previamente. Verifican la firma digital, que cada votante haya votado sólo una vez y que corresponda a un votante válido. Para este esquema se considerará sólo la primera papeleta de votación que contenga una firma válida para un votante, el resto serán rechazados y publicadas las razones. Para aquellas papeletas de votación cuya firma ha sido aceptada se verifica que la ZKP corresponda a la de un voto válido. Posteriormente, se publica en el BB si la papeleta ha sido aceptada o rechazada. Si la papeleta ha sido rechazada, debe publicarse el motivo. Este proceso lo puede revisar cualquier observador, sea interno o no (Verificación Universal).

3.3.6. Suma de votos

Para este proceso las autoridades multiplican los votos encriptados para generar la encriptación del total de los votos, ésta tendrá la forma $E(v_0 M^0 + .. + v_{L-1} M^{L-1}, r)$ donde v_i corresponde al número de votos asociados al candidato i-ésimo y r coresponde a un número aleatorio generado por el votante. Sólo se considerarán dentro de este proceso aquellos votos que fueron aceptados en el proceso de revisión. Cada autoridad hará los cálculos por sí misma con la información que se encuentra publicada. Un voto se considerará válido si dos o más autoridades publicaron su aceptación y se declarará inválido en caso contrario.

3.3.7. Desencriptación de u^F

El valor F corresponde a los resultados de la votación. La desencriptación debe ocurrir por capas y para ésto sólo son necesarias 2 de las 3 autoridades. Éstas deben pertenecer a $Qual$. Cada autoridad posee un *share* x_i que corresponde a un punto en el polinomio cuyo valor en 0 es x, es decir, la clave privada. Si se tiene x la desencriptación ocurre de la siguiente manera:

$$E_x^{-1}(g^r, u^m g^{xr}) = (u^m g^{xr})(g^r)^{-x} = u^m \bmod N^{s+1}$$

Sin embargo, como x se encuentra distribuida, por interpolación de Lagrange se tiene que:

$$x = \sum_{i \in Qual(2)} \left(x_i \prod_{j \in Qual(2), j \neq i} \frac{j}{j-i} \right) \bmod \theta$$

Sean i,j las autoridades seleccionadas para desencriptar los resultados. La autoridad P_i publica $(g^r)^{x_i \frac{j}{j-i}}$ y la autoridad P_j publica $(g^r)^{x_j \frac{i}{i-j}}$. Además de estos últimos valores, cada autoridad debe publicar una *proof* que garantice que los *share* utilizados corresponden a los mismos que obtuvo durante el protocolo DKG. Para demostrar ésto, se utiliza una *proof* de igualdad de logaritmos discretos. Esta *proof* se describe en el cuadro 3.1 mediante un protocolo-Σ. En ella se desea demostrar que los valores y, z corresponden a los valores de p, q elevados a un mismo exponente a respectivamente.

PROVER		**VERIFIER**
0 - Entrada común: $(y,z) = (p^a, q^a)$		
1 - Mensaje inicial: $(\alpha, \beta) = (p^w, q^w)$	$\longrightarrow$	
	$\longleftarrow$	**2 - Desafío:** $e = h(y, z, \alpha, \beta)$
3 - Respuesta: $r = w + ae$	$\longrightarrow$	
		4 - Acepta si: **1**. $p^r = \alpha y^e$ **2**. $q^r = \beta z^e$

Cuadro 3.1: Prueba de igualdad de logarítmos discretos.

Para el caso de la desencriptación, se desea probar que una autoridad P_i, al publicar su desencriptación parcial, utiliza el mismo *share* generado durante el protocolo DKG. La autoridad P_i debe publicar, además, una *proof* de acuerdo a lo descrito en el cuadro 3.1. Con esta *proof* demuestra que $y = (g^{\frac{j}{j-i}})^{x_i}$ y $z = (g^r{}^{\frac{j}{j-i}})^{x_i}$ poseen en común el exponente x_i, que corresponde al *share* de dicha autoridad.

El valor de u^F se obtiene mediante la multiplicación de las desencriptaciones parciales de las dos autoridades encargadas del proceso de desencriptación cuyas *proof* resulten válidas.

3.3.8. Obtención del resultado a partir de u^F

Se basa en lo estudiado en [17]. Dadas las propiedades del anillo escogido y el valor $u = N + 1$ utilizado es posible obtener el resultado. Para ésto, se utiliza inducción. Sea $\tau = u^F \bmod N^{s+1}$. Se desea calcular $F = \mathrm{DLog}_{Z^*_{N^{s+1}}, u}(\tau)$.

Se define la función $D()$ como $D(b) = \frac{b-1}{N}$. Notar que:

$$D((N+1)^U \bmod N^{s+1}) = \sum_{j=1}^{s} \binom{U}{j} N^{j-1} \bmod N^s, \; \forall U \in Z_{N^s}$$

Esta última fórmula se obtiene a partir de la expansión binomial estándar. Para calcular F definimos la siguiente recurrencia:

$$F_i = F \bmod N^i, \; (i = 1, .., s) \tag{3.1}$$

Notemos ahora que:

$$F_i = F_{i-1} + kN^{i-1} \; (\, i \geq 1) \tag{3.2}$$

Lo anterior es por definición de la operación *mod*. Mostraremos ahora cómo calcular nuestro objetivo $F_s = F$ a través de calcular la recurrencia anterior:

- Observemos que F_1 es fácil de calcular y es igual a:

  $$F_1 = F \bmod N = D(u^F \bmod N^2)$$

  Ésto es por la ecuación 3.1.

- Para calcular F_i donde $i > 1$ usamos la ecuaciones 3.1, 3.2 junto a la observación que $\binom{F_j}{t+1} N^t = \binom{F_{j-1}}{t+1} N^t \bmod N^j$, $\forall 0 < t < j$. Con ello se obtiene :

  $$F_i = D(u^F \bmod N^{i+1}) - \sum_{j=2}^{i} \binom{F_{i-1}}{j} N^{j-1} \bmod N^i$$

El valor F_s, en este caso, corresponde al resultado de la votación.

3.3.9. Software del observador

Este proceso monitorea cada una de las etapas realizadas por las autoridades. Su participación se hace presente en los siguientes procesos:

- Durante el proceso inicial de Generación de Parámetros se encarga de corroborar los resultados entregados por las autoridades, e indicar si hay o no irregularidades.

- Durante el protocolo DKG se encarga de verificar aquellos valores calculados por las autoridades que resulten públicos. Es decir, verifica qué autoridades al final del primer paso pertenecen al conjunto *Qual*. A continuación, verifica que aquellas quejas publicadas en el quinto paso sean válidas. Finalmente, se encarga de calcular el valor de aquel polinomio reconstruido en caso de que corresponda y calcular el valor de la clave pública.
- En el proceso de revisión de votos válidos, este proceso verifica que efectivamente sean aceptadas aquellas papeletas de votación consideradas válidas, y en caso de existir irregularidades lo da a conocer.
- Durante la suma de votos comprueba que el valor publicado por cada una de las autoridades sea el correcto de acuerdo a la revisión de votos válidos.
- Durante el proceso de desencriptación, verifica la correctitud de las *proof* publicadas por cada una de las autoridades. Luego, recalcula el valor generado por la desencriptación de acuerdo a las desencriptaciones parciales publicadas por las autoridades encargadas.
- Finalmente, este proceso realiza un cálculo redundante del final de las elecciones y verifica que los resultados concuerden con los calculados por las autoridades.

3.4. Detalle de la implementación

La implementación se llevó a cabo utilizando el lenguaje Java (JDK V1.6). Utilizando las librerías criptográficas provistas por *java.security* y el mecanismo para invocar un método remotamente RMI. Los votos y los datos enviados vía RMI son serializados a un formato XML antes de ser enviados y luego son publicados por el BulletinBoard en este mismo formato.

3.4.1. Módulos implementados

1. **Generación de claves de los votantes (GeneraClaves.java):** Por motivos prácticos, para este sistema las claves de los votantes se almacenarán en un archivo dentro del mismo computador, sin embargo, en la práctica la clave privada debe ser almacenada en un dispositivo seguro y no en las DRE. En este módulo se genera un par de claves para cada votante que corresponden a la clave pública y privada del esquema de firmas digitales RSA. Se entrega como entrada el parámetro de seguridad k para la generación del módulo RSA. Las claves públicas son enviadas al BulletinBoard serializadas a un formato XML y posteriormente publicadas por el mismo para que más tarde las autoridades puedan solicitarlas en la verificación de las papeletas de votación. Algunas de las funciones utilizadas en este módulo son las siguientes:
   - public void initialize(int keysize);
     Inicializa el generador de claves para un largo de clave determinado.

- public KeyPair genKeyPair();
  Se utiliza para obtener las claves públicas y privadas que serán utilizadas por los votantes para firmar sus votos.

Estas funciones pertenecen a la clase java.security.KeyPairGenerator.

2. **Generación de parámetros iniciales Generador.java** Este módulo utiliza básicamente 3 funciones. Éstas son:

   - public BigInteger BGWpub(BigInteger pi, BigInteger qi)
     Esta función retorna el cálculo final de N entregado por el protocolo BGW.
   - public BigInteger BGWdist(BigInteger pi, BigInteger qi)
     Esta función retorna un *sharing* aditivo del valor compartido por las autoridades en cada iteración del protocolo de transformación de un *sharing* multiplicativo en un *sharing* aditivo descrito en la sección 2.8.
   - public boolean FTIE(BigInteger g, BigInteger N)
     Esta función es la implementación del paso 3 descrito en la sección 2.8. Su retorno indica si N corresponde o no a la multiplicación de dos primos de largo suficiente. Si esta función retorna *false* para 2 o más autoridades se debe repetir nuevamente todo el proceso de generación de N.

   Cada una de estas funciones requieren comunicación directa con el BulletinBoard para obtener lo datos publicados por las demás autoridades. Los valores publicados por cada una de ellas son enviados con la firma de la autoridad al BulletinBoard en formato XML vía RMI.

3. **Protocolo DKG (Autoridad.java):** En este módulo las autoridades generan su par de claves RSA para la encriptación y el uso de firmas digitales y la envían al BulletinBoard que posteriormente las publica. Una vez generados estos valores, comienzan la ejecución del protocolo DKG. Las autoridades envían paso a paso los valores generados al BulletinBoard y luego reciben como respuesta los valores publicados por las demás autoridades. Se siguen los siguientes pasos una vez generadas las claves y publicadas en el BulletinBoard:

   - Paso 1: Se generan los dos polinomios descritos en 2.6 de manera aleatoria. Para lo que se genera el par de coeficientes aleatorios correspondientes a cada uno de los polinomios. En este proceso se utiliza la clase SecureRandom para generar un random y posteriormente se genera un BigInteger aleatorio mediante el siguiente constructor:
     - BigInteger(int numBits, Random rnd): Genera un BigInteger aleatorio en el rango $[1, 2^{numBits} - 1]$.

     Luego, cada autoridad calcula y publica los valores C_{ik} correspondientes y los valores s_{ij} (ver sección 2.6). Estos últimos valores van encriptados para la autoridad j correspondiente utilizando RSA. Una vez publicados los valores se recibe respuesta de las otras dos autoridades, estos valores se reciben en formato XML y vienen junto a una firma digital de la autoridad que lo envía.

- Paso 2: Cada autoridad verifica que los valores recibidos por las otras autoridades contengan una firma válida. Para ésto, solicita al BulletinBoard la clave pública de cada autoridad. En este módulo se utiliza la clase SignedObject.java y la siguiente función:
  - boolean verify(PublicKey verificationKey, Signature verificationEngine): Verifica que la firma en el SignedObject es válida para el objeto que contiene con la clave pública utilizando el algoritmo de verificación correspondiente.

  Si la firma corresponde, se verifican los valores enviados por cada autoridad según los cálculos definidos en la sección 2.6. En seguida, la autoridad envía un arreglo de valores en los que indica si hay o no quejas para cada autoridad (los valores van firmados). Luego, recibe los valores publicados por las otras autoridades.
- Paso 3: Se verifican las firmas de los valores recibidos y posteriormente se genera el conjunto de las autoridades pertenecientes a *qual* verificando que no presenten más de una queja válida. De lo contrario, son automáticamente descartadas.
- Paso 4: En este paso las autoridades generan su *share* de acuerdo a lo descrito en la sección 2.6 y la almacenan en su memoria. Luego, publican en el BulletinBoard los valores A_{ik} y reciben los valores publicados por las demás autoridades.
- Paso 5: Se verifican las firmas y luego los valores recibidos para cada autoridad de acuerdo a la sección 2.6: Ecuación 2.2 y en caso de no cumplirse dicha ecuación la autoridad publica los valores recibidos en el paso uno por dicha autoridad. En seguida, recibe los valores publicados por las demás autoridades.
- Paso 6: Se verifican las firmas y luego se verifican los valores publicados en el paso anterior. Si realmente dichos valores respetan la ecuación 2.1 y no la ecuación 2.2 definida en la sección 2.6, aquella autoridad perteneciente a *qual* contra la cual se ha publicado una queja se considera deshonesta y las demás autoridades proceden a reconstruir su polinomio. Para ésto, cada autoridad inicia su reconstrucción mediante el polinomio de Lagrange y publican el trozo calculado en el BulletinBoard.
- Paso 7: Las autoridades pertenecientes a *qual* consideradas honestas en el paso 6 realizan los cálculos de reconstrucción del polinomio de aquella autoridad deshonesta y publican sus valores en el BulletinBoard.
- Paso 8: Cada autoridad perteneciente a *qual* considerada honesta en el paso 6 calcula la clave pública y la publica en el BulletinBoard.

Las salidas privadas del protocolo son almacenadas por cada autoridad en un archivo, para más adelante ser utilizados en la desencriptación.

4. **Generación del voto, ZKP y firma (Main.java)**

   Para la encriptación de los votos se utiliza la clase BigInteger. Para la generación de números *random* seguros se utiliza la clase SecureRandom. Además, para el cálculo del *hash e* se utiliza la clase MessageDigest.

   Se implementaron dos clases adicionales a la clase principal (Main). Éstas son:

   - **ZKP:** Un objeto de esta clase contiene los siguientes elementos:

BigInteger [] E=new BigInteger[2];
BigInteger [] ci;
BigInteger [] c_i;
BigInteger [] c_di;
BigInteger [] c_dib;
BigInteger cy;
BigInteger [] Ey=new BigInteger[2];
BigInteger [] fi;
BigInteger [] z_1i;
BigInteger [] z_2i;
BigInteger z_3;
BigInteger z_4;
BigInteger D;

Estos valores pueden verse en detalle en el cuadro 2.4. Según ella, los valores E, ci, c_i corresponden a la entrada común, los valores c_di, c_dib, cy, Ey corresponden al commitment inicial, el valor e corresponde al desafío y los valores fi, z_1i, z_2i, z_3, z_4, D corresponden a la respuesta al desafío. Para obtener el valor e se utiliza el algoritmo de *hash* $SHA1$, éste se aplica sobre los valores iniciales y los valores generados en la primera etapa del protocolo-Σ, es decir, es el desafío. Como se mencionó anteriormente, la razón por la que ésto puede hacerse es porque al ser el *hash* una función unidireccional un posible adversario no podría jugar con los valores para generar un desafío de su conveniencia de manera sencilla, ya que un mínimo cambio en los valores iniciales del protocolo alteraría todo el resultado del desafío, y a partir de un valor en el desafío es muy difícil encontrar la entrada que generó ese *hash*. En la ZKP se incluye el voto encriptado. Por lo tanto, sólo queda generar la firma.

- **votocompleto:** Esta clase contiene dos valores, que son:

  ZKP proof;
  SignedObject so;

  El primero, como se explicó anteriormente, contiene el voto encriptado y la ZKP. El segundo, contiene la firma digital DSA del objeto anterior utilizando la clave pública del votante.

- **Main:** El votante ingresa su elección que es un vector que contiene L espacios. De éstos, se marcarán con 1 aquellos que representan la elección del votante y en 0 aquellos que no pertenezcan al conjunto de los ρ elegidos. En la implementación se utilizan las siguientes funciones:

  - public ZKP(int [] eleccion, parametros par);
    Al construir un objeto de esta clase con estos parámetros se genera la ZKP y el voto encriptado.
  - public SignedObject(Serializable object, PrivateKey signingKey, Signature signingEngine);
    Clase provista por java.security, para generar firmas digitales.

- public votocompleto(proof, so);
  Acá se asignan los valores para crear el elemento.

Una vez generado el voto se utiliza las librerías de Xstream para serializar este objeto a XML y enviarlo al BulletinBoard para su publicación.

5. **Revisión de votos válidos, suma de votos y desencriptación del total (revision.java)**
   Para la revisión de votos válidos se utilizan principalmente dos funciones:

   - public boolean verificar(ZKP proof, parametros par);
     Se basa en la etapa de verificación del cuadro 2.4, si corresponde entrega *true* o $false$.
   - public boolean verify(PublicKey verificationKey, Signature verificationEngine);
     Esta función se ocupa para verificar si la firma digital corresponde realmente a la firma del votante, provista por la clase SignedObject.

   Inicialmente, cada autoridad solicita al BulletinBoard las papeletas de votación generadas por todos los votantes que participaron en el proceso de las elecciones. Cada autoridad verifica los votos uno a uno; primero, verificando que la firma corresponda a quien votó. Luego, verificando que la *proof* generada sea válida y posteriormente verificando que el mismo votante no haya generado anteriormente un voto válido. Si un voto es rechazado por una autoridad, ésta publica el motivo del rechazo. Después de haber hecho los cálculos, cada autoridad publica en el BulletinBoard en formato XML los votos que han sido aceptados o rechazados y los motivos de rechazo. Al publicar se recibe los valores publicados por las otras autoridades.

   Cada autoridad calcula la encriptación de la suma de los votos válidos, y para ésto, sólo considera aquellos votos que hayan sido aceptados por al menos dos autoridades. Luego, cada autoridad publica en el BulletinBoard este valor.

   Se consideran sólo dos autoridades honestas y estas envían su desencriptación parcial y una *proof* para el valor utilizado (sección 3.3.7) al BulletinBoard. A continuación, cada autoridad verifica las *proof* y calcula la desencriptación total a partir de los valores enviados y publica su cálculo de u^f en el BulletinBoard.

   Finalmente, cada autoridad obtiene el resultado de la votación en el formato definido en 2.2.3 mediante la siguiente función:

   - BigInteger obtenerTotal(BigInteger um, BigInteger n, int s)

   Luego publica este valor en el BulletinBoard.

6. **BulletinBoard (elementoImpl.java)** El BulletinBoard se encarga de recibir valores y publicarlos. También es el encargado de entregar los valores que se le soliciten ya que todo valor publicado en el BulletinBoard es de conocimiento público. Las siguientes son las funciones que implementa:

   - public elementoImpl(): Constructor, permite generar un objeto sobre el que trabajarán todas las autoridades mediante el uso de RMI

- public String [] publicaCI(String envio): Mediante esta función las autoridades publican los valores que van generando a lo largo del DKG y desencriptación del total y reciben como respuesta los valores publicados por las demás autoridades.
- public int publicaPK(String envio1): Al llamar a esta función se publica la clave pública. El String envio1 debe venir en formato XML según una clase previamente definida indicando el identificador y la clave pública de quien los envía. Si la respuesta es 0 es porque el valor no fue publicado, ya sea porque ya existía una clave pública para esa id o porque el formato del valor enviado no corresponde.
- public String solicitaPK(int id): Esta función se utiliza para solicitar la clave pública correspondiente a una cierta id. Se retorna un String con la clave pública solicitada en formato XML.
- public int publicavoto(String voto): Mediante esta función los votantes publican su papeleta de votación en el BulletinBoard. Se retorna un 1 si la papeleta fue publicada exitosamente y un 0 si el formato del elemento enviado no corresponde al esperado.
- public String solicitavotos(): Mediante esta función las autoridades u observadores externos solicitan el conjunto de papeletas de votación emitidas por los votantes.

7. Sumado a las clases indicadas anteriormente, se utilizan clases auxiliares adicionales, que poseen una determinada estructura para hacer las transformaciones desde y hacia el formato XML. Estas clases y su descripción son las siguientes:

   - validación.java: Un objeto de esta clase contiene el estado de una papeleta de votación y su estado. Posee los siguientes elementos:
     - String id: Identificador de la papeleta.
     - estado: El estado de una papeleta puede ser 1 si es aceptada ó 0 si es rechazada;
     - String motivo: Si una papeleta fue rechazada se indica el motivo, este puede ser "Objeto no válido", "firma rechazada", "ZKP rechazada" o "Ya ha votado".
     - validacion sig: Este elemento se usa para generar una estructura de nodos con la validación de cada uno de los votos y luego generar un XML del conjunto de validaciones.
   - sk.java: Esta estructura la utiliza cada autoridad para publicar los valores secretos que deba almacenar en memoria, éstos van dentro de un archivo y se solicitan mediante una función. Contiene los siguientes elementos:
     - String tipo: Indica el tipo de valor almacenado, por ejemplo "share", "SK" etc.
     - String valor: Contiene el valor correspondiente para ese tipo en formato XML.
   - pk.java: Se utiliza para el envío de claves públicas a traves de RMI. Está formado por los diguientes elementos:
     - int ids: Contiene el identificador del dueño de la clave pública.
     - String pks: Contiene la clave pública en formato XML.
   - enviosDKG.java: Utilizada para el envío de valores durante el DKG. Un objeto de esta clase posee los siguientes elementos:

- int autoridad: Indica el identificador de la autoridad que envía el valor.
- String publicos: Contiene el objeto que se envía junto a su firma digital.
- int paso: Indica el paso del protocolo que se está publicando.

8. Por último, para generar la comunicación via RMI se utilizan clases adicionales que sirven para permitir la comunicación remota, éstas son:

   - elemento.java: Interfaz que extiende a la clase Remote. Cuya implementación se realiza en elementoImpl.java.
   - Bb.java: Crea el objeto remoto para el envío de mensajes mediante RMI.

9. **Software del observador (Obs.java)** Este *software* se puede ejecutar luego de cada proceso en el que estén involucradas las autoridades. Éstos son los procesos de generación de parámetros iniciales (0), protocolo DKG (1), revisión de votos válidos y suma de votos (2), desencriptación (3) y cálculo final (4). Se ejecuta entregando como parámetro el número del proceso que será analizado (indicado anteriormente entre paréntesis). En cada proceso se entrega lo siguiente:

   - Generación de parámetros: Recalcula N de acuerdo a los valores publicados por las autoridades e imprime si corresponde al valor publicado en el BB y que este valor corresponda a la multiplicación de dos primos de largo suficiente. Comprueba que los aleatorios utilizados correspondan a los generados por el *hash* del valor previamente acordado.
   - Protocolo DKG: Verifica los pasos 5 y 6 del protocolo y recalcula el resultado final de la clave pública y.
   - Revisión de votos válidos y suma de votos: Revisa papeleta a papeleta si son o no válidas y luego publica la encriptación del total, comparándola con los valores obtenidos desde el BulletinBoard.
   - Desencriptación: Verifica las *proof* emitidas por las autoridades involucradas e imprime quejas si los valores no corresponden.
   - Cálculo final: Calcula e imprime el resultado de las elecciones mediante un cálculo independiente al de las autoridades.

   Durante este proceso los valores publicados previamente por las autoridades son consultados al BulletinBoard. Los valores son recibidos vía RMI en formato XML y posteriormente transformados a su formato original.

Capítulo 4

Extensiones y variantes

Se proponen a continuación posibles extensiones y variantes del sistema desarrollado, que incluyen seguridad a nivel de *hardware* y *software*.

4.1. Arquitectura verificable para el software del votante

A continuación, se explicarán técnicas para verificar propiedades de seguridad en máquinas de votación electrónica DRE basado en lo estudiado en [27]. Se deben considerar dos propiedades principales, éstas son:

- Propiedad 1: Ninguna interacción del votante con la máquina puede afectar la interacción de un votante anterior.
- Propiedad 2: Un voto no puede ser contado sin el consentimiento del usuario.

Se considera que los usuarios son lo suficientemente atentos como para tomar las medidas de seguridad que a ellos les corresponde tomar, y que saben llevar a cabo el proceso de votación en las DRE.

El problema actualmente en las DRE es que el código base a confiar (TCB o trusted computing base) es muy largo, ésto hace que sea muy difícil verificar que el código no contenga líneas maliciosas. La idea es reducir el código necesario a confiar al máximo, para que así el verificar la confiabilidad del sistema sea un proceso más sencillo. Se explicará a continuación una manera de hacerlo.

Los sistemas de votación dividen el proceso de elección del votante en dos partes: En la primera, el votante selecciona a un candidato, en la segunda, confirma su voto. La

idea de esta arquitectura es separar ambos procesos protegiendo su integridad mediante técnicas de aislación de *hardware*, además, propone verificar sólo que la fase de confirmación se realice correctamente, reduciendo así significativamnte el TCB. Con las técnicas apropiadas, utilizando lo anterior se puede demostrar que un sistema cumple con la propiedad 2.

Para asegurar el cumplimiento de la propiedad 1 se propone utilizar reseteo de *hardware*, para restaurar el estado de sus componentes y así asegurar que cada usuario será tratado de la misma manera, eliminando riesgos de privacidad.

Este estudio se centra en la fase de votación activa, es decir, el votante recibe su *token* (por ejemplo, una tarjeta inteligente) una vez que se autentifica, el DRE verifica que el *token* es correcto, el votante selecciona y confirma su voto, el voto es almacenado y el *token* es deshabilitado.

El problema a resolver consiste en cómo asegurar que el diseño original y la implementación sean seguros.

4.1.1. Detalle de la arquitectura

Luego de que un votante ha confirmado su voto la máquina debe reiniciarse. Los archivos almacenados se dividen de dos maneras. Por una parte, se encuentran los archivos de configuración que sólo permiten su lectura y, por otra parte, aquellos archivos que sólo permiten su escritura que corresponden a aquellos votos antes realizados, por ejemplo. De todos modos debe haber alguna forma de asegurar que no se sobreescriba los datos existentes ya que por el momento sólo se está protegiendo su lectura. Para ésto, se introduce un módulo separado, cuyo fin es manejar el proceso de reseteo. Debe ser llevado a cabo un proceso de aislación del proceso de confirmación, ya que de él dependerá la confiabilidad del sistema.

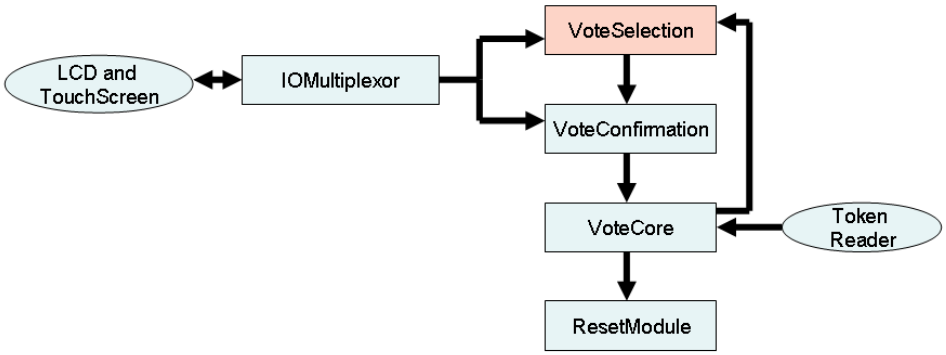

Figura 4.1: Arquitectura verificable para el software del votante

Inicialmente el votante se encuentra en el módulo VoteSelection, luego de seleccionar pasa al módulo VoteConfirmation. El módulo de confirmación se encarga de mostrar al usuario la opción antes escogida y éste debe confirmar si corresponde o no a su elección previamente

realizada. Ambos módulos mencionados anteriormente se comunican con un módulo llamado IOMultiplexor, y éste se comunica a su vez con el LCDAndTouchScreen. Se debe asegurar que durante el proceso de confirmación sea el módulo VoteConfirmation quien se esté comunicando con IOMultiplexor, y no VoteSelection. Para que ésto se realice correctamente, el módulo VoteConfirmation tiene preferencia del LCD, por lo tanto, una vez que lo solicita el VoteSetection queda bloqueado, y sólo a VoteConfirmation se le otorga la comunicación con IOMultiplexor. El módulo IOMultiplexor debe ser confiable. Debe existir, además, un módulo llamado VoteCore el que se lleva a cabo una vez terminado el módulo de VoteConfirmation. El módulo VoteCore se encarga de almacenar el voto en el BallotBox y en seguida inhabilitar el *token*. Es, también, este módulo el encargado de verificar el *token* al inicio, si es correcto y enviar un mensaje al módulo VoteSelection para que comience a funcionar. Los procesos en VoteCore se realizan de manera atómica, de manera de que no será receteado antes que todos ellos se hayan completado. Una vez completas todas estas acciones el VoteCore envía una señal al ResetModule.

La separación de los módulos anteriormente mencionados se debe llevar a cabo a nivel de *hardware* (Se utiliza aislamiento de *hardware* para garantizar que el aislamiento sea más fuerte. Se ejecuta cada módulo en su propio microprocesador, es decir, con su propia CPU, RAM e Interfaces I/O).

La arquitectura a nivel de *hardware* ocupa dos tipos de canales de comunicación: Buses y *wires*. Los buses permiten comunicación unidireccional o bidireccional de alta velocidad entre los distintos módulos o componentes y los *wires* son elementos que permiten emitir señales con un simple bit de estado. Estas señales, pueden ser altas o bajas y en este caso se usarán para indicar la presencia o ausencia de algún evento. Se asume que los *wires* son confiables pero los buses pueden no serlo.

En esta arquitectura, cuando una componente desea transmitir un mensaje lo enviará repetitivamente por el bus hasta que sea reseteada o reciba una señal para dejar de transmitir. Quien emite anexa un *hash* del mensaje y lo envía. El receptor acepta el primer mensaje con un *hash* válido y luego acusa recibo mediante una señal. Estas señales se pueden realizar cambiando el valor de un *wire* de alto a bajo.

El proceso comienza cuando el VoteCore recibe un *token* válido. Posteriormente, envía los datos del *token* al módulo VoteSelection hasta que reciba un mensaje de VoteConfirmation. Después de almacenar el voto y de anular el *token* el módulo VoteCore gatilla el reseteo colocando su *wire* hacia el ResetModule en alto. Para comunicarse con el votante voteSelection genera un mapa de bits de una imagen, la empaqueta en un mensaje y lo envía a IOMultiplexor repetitivamente. Cada imagen va anexada a un número que no cambia si la imagen se mantiene constante. El módulo IOMultiplexor leerá repetitivamente sus mensajes, inicialmente provenientes del módulo VoteSelection y será el encargado de que sean desplegados en pantalla todos aquellos que traigan anexados un número distinto. El IOMultiplexor, además, interpreta entradas desde el *touchscreen*, determinando cuando las entradas corresponden a un botón virtual y si es así, repetitivamente escribe el nombre de la región del módulo VoteSelection hasta que tenga un nuevo input del votante. El nombrar las regiones previene de que los input del votante en una pantalla sean interpretados como entradas en otra pantalla.

Cuando un votante decide pasar desde la etapa de seleccción a la etapa de confirmación, el módulo VoteConfirmation recibirá un voto proveniente del módulo VoteSelection. El módulo VoteConfirmation seteará su *wire* con IOMultiplexor en alto. Cuando IOMultiplexor detecta el *wire* en alto, vaciará todos sus buffer de entradas y salidas, reseteará su contador para los mensajes de VoteSelection. Posteriormente, manipulará las entradas y salidas sólo para el módulo de VoteConfirmation. Si el módulo VoteConfirmation determina que el votante quiere regresar al módulo VoteSelection para editar su voto, entonces seteara su *wire* con el módulo VoteSelection en alto. Luego, el módulo VoteSelection usará su bus conectado al módulo VoteConfirmation para saber que es su turno para trabajar. El módulo VoteSelectión recibe el mensaje y el módulo VoteConfirmation es reseteado. Éste setea sus *wire* con IOMultiplexor y VoteSelection en bajo. Una vez detectando ésto, el IOMultiplexor borrará todas sus entradas y salidas como lo hizo anteriormente y volverá a recibir entradas y salidas del módulo de VoteSelection.

4.1.2. Análisis de la arquitectura

Este tipo de arquitectura permite reducir el espacio de código a confiar, ya que sólo se debe verificar que los módulos de IOMultiplexor, VoteConfirmation, VoteCore y ResetModule sean confiables. Independiente de si en el resto de los módulos se encuentra código malicioso, éste no podrá llevarse a cabo dada la forma en que está estructurado el proceso a nivel de *software* y *hardware*. Este tipo de arquitectura garantiza el cumplimiento de la propiedad 1 y 2 de una manera más sencilla, por lo tanto, es una manera interesante de comprobar una vez realizado el código si éste ha sido desarrollado de la manera correcta sin necesidad de revisarlo completamente.

4.2. Implementación del BulletinBoard como un Broadcast

El BulletinBoard se podría eliminar de existir un *broadcast*, es decir, una entidad que a penas recibe un valor lo envía a todo los participantes, que en este caso son las autoridades y el observador.

Un protocolo de *broadcast* debe cumplir con dos propiedades, éstas son:

- **Correctitud:** Dado un enviador D honesto. Si D envía un mensaje m mediante el protocolo de *broadcast*, todos los receptores honestos recibirán m.
- **Consistencia:** Al final del protocolo de *broadcast* todos los receptores honestos reciben el mismo valor.

La primera propiedad permite garantizar que a pesar de la presencia de ciertos receptores maliciosos el valor enviado m será recibido de manera íntegra por todos los receptores honestos.

La segunda propiedad permite garantizar que a pesar de la coalisión de un enviador corrupto con receptores corruptos los receptores honestos siempre recibirán el mismo valor. Para que se cumpla esto último, deben existir condiciones previamente establecidas para acordar un valor específico en ciertas situaciones críticas.

Actualmente, el BulletinBoard recibe un valor y lo almacena en su memoria. Si un participante consulta un valor al BulletinBoard, éste envía el valor consultado en respuesta de manera remota. Mediante el reemplazo del BulletinBoard con un *broadcast* se tendría un escenario diferente. Inicialmente el enviador comparte un valor con los demás participantes. Este valor es enviado mediante un protocolo de *broadcast*. Una vez recibido este valor por cada uno de los participantes éstos interactúan entre sí. Mediante esta serie de interacciones se garantiza el cumplimiento de las dos propiedades previamente mencionadas. Una vez llegado a un acuerdo cada una de las partes involucradas almacenan el valor recibido en su memoria local. Finalmente, las operaciones de consulta se hacen directamente en la memoria local del receptor y no de manera remota como ocurre actualmente.

Al reemplazar el BulletinBoard por un *broadcast* se requiere de uno o más participantes adicionales involucrados en el protocolo de *broadcast*. Este participante adicional es el observador. El observador permite a una entidad externa seguir cada paso en la votación electrónica y así obtener verificación universal. Una buena forma de hacer ésto, es tener varios observadores (servidores) corriendo en distintos lugares independientes entre sí (por ejemplo, cada observador en un partido político distinto). Ésto permite que la confianza no se encuentre centralizada, sino distribuida entre distintas partes.

4.3. Extensiones orientadas a disminuir la coerción

Existen algunas propuestas que permiten reducir los casos de coerción, éstos son una de las principales preocupaciones que envuelven a la votación electrónica. En [28] se propone el uso de votos reemplazables para prevenir la coerción. Este esquema permite al votante cambiar su voto, y dado que utiliza *mix-nets* previo a su publicación, un adversario (quien está ejerciendo coerción sobre el votante), no podría saber si a futuro este voto fue o no cambiado, ya que al pasar por las *mix-nets* se perdería la huella del voto. Las *mix-nets* en cada paso publican *proofs* que garantizan que las operaciones que están llevando a cabo son las correctas.

En este esquema se usan boletos de voto dinámicos . En ellos el orden de los candidatos en el boleto es creado dinámicamente y cambia para cada votante. Este esquema está orientado a redes, como por ejemplo Internet. Los autores de este documento ([28]) aseguran satisfacer propiedades como la incoercibilidad y la privacidad, que son difíciles de controlar en una red. El voto dinámico está orientado a prevenir que un adversario que monitorea lo que el usuario está haciendo (*hacker*) pueda sacar información respecto a sus elecciones.

La posibilidad de votar nuevamente reemplazando el o los votos anteriores se propone como clave para la prevención de los casos de coerción en la votación por Internet ya que

nadie puede saber si un voto será el definitivo o no. Este punto es discutible, ya que aún pueden existir problemas de coerción como, por ejemplo, forzar a una persona a votar a finales del periodo electoral y luego asegurarse de que éste no tenga acceso a un computador hasta que se hayan cerrado los procesos electorales.

Capítulo 5

Desafíos en la práctica

5.1. Necesidad de PKI

Es absolutamente necesario en un sistema de votación electrónica el uso de PKI para la autentificación de los votantes. Es decir, para garantizar mediante firmas digitales, que un voto fue generado realmente por quien dice haberlo hecho. Una de las principales razones para ésto es la verificación universal. Se mencionó en 1.6.1 que una de las ventajas de los esquemas de clave pública es que sólo se necesita un par de claves por cada firmante A, y basta tener la clave pública de A para verificar la correctitud de una firma que le pertenezca. En este caso, cada votante debe tener un par (PK, SK) que debe ser generado previo a los procesos de validación, votación y escrutinio. Lo mismo para cada autoridad involucrada y observador en caso de implementar el BulletinBoard como un *broadcast*. Estas firmas, además, deben estar acompañadas de un certificado digital confiable[1] que garantice que realmente una clave pública pertenece a quien dice hacerlo.

En la práctica las autoridades pueden utilizar un sistema de certificación cruzada y los votantes ser certificados por las tres autoridades.

5.1.1. Uso de smartcards

Existen propuestas respecto a la generación y uso de PKIs en la práctica. Una de ellas es el uso de *smartcards* o tarjetas inteligentes.

Las *smartcards* son dispositivos del tamaño de una tarjeta de crédito que contienen un *chip* con la clave privada. Todos los procesos computacionales de firma pueden ser realizados dentro de la tarjeta para que así la clave privada no deba ser entregada a otro *hardware*.

[1] Generados por una autoridad certificadora conocida y confiable.

Se requiere un centro de creación de *smartcards* donde los dispositivos sean personalizados para los votantes y una autoridad certificadora que controle la certificación de las claves para cada votante. Se podría utilizar, también, un PIN adicional para identificar a un votante con su Smartcard, que debe ser entregado de manera independiente al votante.

5.2. Ataques de denegación de servicios (DOS) y conectividad

Los ataques de denegación de servicio son los más comunes ya que, por lo general, son los más fáciles de llevar a cabo. Existen distintos métodos para realizar ataques de denegación de servicios.

Una forma de realizarlos es llenar la red de datos sin sentido impidiendo que aquellos legítimos puedan pasar a través de ésta. Otro tipo de ataque es colmar al servidor con tareas inútiles que hagan que ocupe por completo sus recursos impidiendo que los usuarios reales puedan hacer uso de éste. La intervención es aun peor si se trata de un ataque DDOS (ataque de denegación de servicios distribuido), en el que colaboran varias máquinas para lograr en conjunto el objetivo. Ésto puede lograrse, en primer lugar, tomando el control de muchas computadoras de antemano, a través de un mecanismo de envío como, por ejemplo, virus o gusanos. Una vez infectados, los ordenadores se convierten en "esclavos" que cumplen las órdenes que envía el equipo principal, lo que permite al atacante cumplir su objetivo con un envío simultáneo de datos o solicitudes. Se deben considerar este tipo de ataques en aquellas máquinas que requieren conexión de redes.

Con la tecnología actual es difícil impedir que un determinado ataque DOS se lleve a cabo, ya que una vez que se ha puesto en marcha, la forma de detenerlo es el cierre de todas las comunicaciones. Por lo tanto, deben tomarse las medidas necesarias ante estos casos, por ejemplo, utilizando servidores de reemplazo, respaldo de información, entre otros.

La facilidad con que se pueden realizar los ataques de DOS hace que sean bastante comunes. En Canadá hubo en promedio cuatro mil ataques DOS en la Internet cada semana cuando se permitió el voto remoto por Internet[2]. Miles de votantes no pudieron iniciar la sesión y emitir su voto debido a un ataque DOS que paralizó el servidor central[3]. A pesar de que esta interrupción duró menos de una hora, y que no causó más que molestias y frustraciones, si se logra un ataque DOS persistente durante todo un día de elecciones podría privar a millones de votantes de participar en el proceso electoral. Un amplio periodo de votación podría ayudar a reducir los problemas que causan estos ataques. Sin embargo, no puede eliminar la posibilidad de que afecten a una gran parte de los votantes, ya que la experiencia ha demostrado la tendencia común de esperar hasta el último día para votar.

Existen propuestas para algoritmos de ruteo para prevenir ciertos ataques de denega-

[2]Como una opción de votación en el 2003 para la convención de liderazgo NDP (National Democratic Party).

[3]"Computer Vandal Delays Leadership vote Canadian Broadcasting Corporation". Jan 25, 2003.

ción de servicios. Sin embargo, es probable que la solución definitiva tome tiempo en estandarizarse y masificarse. Por ende, se recomienda usar redes dedicadas (no públicas) para la comunicación en un sistema de votación electrónica.

Capítulo 6

Conclusiones

En base a los objetivos se puede concluir que:

- Mediante el uso del módulo *SO*, definido previamente, es posible para cualquier ciudadano verificar que cada proceso se está llevando a cabo correctamente. Ésto garantiza la propiedad de "Verificación Universal". Este módulo puede ser ejecutado por cualquier entidad, sea interna o externa al proceso, funciona de manera independiente y verifica de manera redundante cada uno de los pasos llevados a cabo por las autoridades. Lo anterior permite corroborar el buen funcionamiento de ellas e informar posibles fraudes. Puntualmente, verificar que cualquier fraude no pase desapercibido.
- Los votos son encriptados utilizando una vaiación del modelo de encriptación ElGamal, el cual conserva sus propiedades. Su seguridad de basa en el problema DDH, cuya solución en nuestros días es infactible. Por lo tanto, a pesar de existir relación entre un voto encriptado y un votante, es infactible aun en presencia de una autoridad comprometida, conocer los candidatos por los cuales el votante votó.
- El esquema se implementó basado en lo propuesto en [15]. Integrando módulos adicionales, propuestos por expertos, que incrementan su seguridad. Éstos son, la generación de parámetros, en particular, del módulo N de manera distribuida. Además, la generación de claves distribuidas mediante el uso del protocolo DKG. Estos dos módulos, evitan depositar la confianza en una sóla entidad, y permiten un factor de error adicional, bajo un fuerte argumento: Se permite, incluso, que una de las autoridades se encuentre comprometida, garantizando que a pesar de que ésta utilice todos sus recursos en quebrar el sistema no lo logrará.
- El esquema implementado reduce la cantidad de operaciones costosas en tiempo de ejecución (exponenciaciones). Esto hace que el proceso de verificación de las ZKP, uno de los más costosos, sea significativamente más eficiente en comparación a los otros esquemas estudiados.

El sistema debe permitir al votante la opción de votar nulo o blanco cuando $\rho = 1$. La razón de ésto es porque, debido al diseño de las ZKP, se podría generar una *proof* que inclu-

ya como elecciones alguna de las opciones anteriores y candidatos adicionales, permitiendo así inconsistencias en la suma final.

Implementar un esquema de votación electrónica seguro no lo es todo. Se deben establecer estándares para los procesos externos, como por ejemplo, que garanticen que las máquinas no han sido modificadas hasta llegar a su destino. En otras palabras, que el *software* que fue implementado bajo estrictas medidas de seguridad sea el mismo que está corriendo en la máquina final. También se deben establecer estándares para los procesos de registro y validación, que garanticen que aquel que emite una papeleta de votación es quien tiene derecho a voto. En el caso de traslado de información en red se debe establecer medidas de seguridad que permitan garantizar una conexión segura. Ésto podría llevarse a cabo, por ejemplo, mediante el buen uso de criptografía y redundancia.

La votación electrónica es una herramienta muy fuerte si se utiliza correctamente. Es decir, si se logra que ésta respete cada una de sus carácterísticas previamente mencionadas. Sus desventajas, en general, se deben a la confianza y familiaridad del ciudadano promedio con el actual sistema de votación en papel. De todos modos, ésto podría ir cambiando con el tiempo, si se logra inculcar en las personas los beneficios que la votación electrónica otorgaría. Sin embargo, lo anterior no es un tema trivial. Una de las tareas más dificiles al llevar la votación electrónica a la práctica, es capturar la confianza del ciudadano en el sistema, es decir, convencer al perdedor que realmente ha perdido. Por que si bien para personas expertas la verificación universal da fe de un buen funcionamiento, para una persona no inmersa en el sistema y en ciencias criptográficas esto puede resultar dificil de creer.

Actualmente, existen aquellos que rechazan completamente su implementación. Los argumentos que llevan a ésto son, por ejemplo, que la necesidad de un cambio se vé cuando el sistema actual no funciona bien. Sin embargo, no todo es así. Un cambio debe llevarse a cabo cuando éste garantiza ser tanto o mejor que el anterior. En el caso de la votación electrónica, ésta permitiría reducir las tasas de errores, por un lado, dando la oportunidad al votante de cambiar un voto que será considerado nulo. Y por otro, reduciendo los errores en el proceso de escrutinio. Otro gran beneficio que implicaría su implementación es la reducción de costo (amortizado) lo que permitiría aumentar la participación ciudadana con elecciones más frecuentes.

La votación vía Internet, si bien en un mundo ideal traería muchos beneficios de accesibilidad que la votación presencial no tiene, actualmente es infactible. Sus dos principales razones, son el no poder garantizar la privacidad y el simple hecho que en nuestros días no todos tienen acceso a Internet.

Bibliografía

[1] Oded Goldreich. *Foundations of Cryptography*, volume Basic Tools. Cambridge University Press, 2001.

[2] Shamir. How to share a secret. *CACM: Communications of the ACM*, 22, 1979.

[3] Goldwasser, Micali, and Rackoff. The knowledge complexity of interactive proof systems. *SICOMP: SIAM Journal on Computing*, 18, 1989.

[4] Burmester. Homomorphisms of secret sharing schemes: A tool for verifiable signature sharing. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1996.

[5] ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEETIT: IEEE Transactions on Information Theory*, 31, 1985.

[6] Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1999.

[7] B. Harris and D. Allen. Black box voting: Vote tampering in the 21st century, 2003.

[8] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system, 2004.

[9] Philip E. Vote early, vote often, and votehere, 2001.

[10] David Chaum. E-voting: Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, pages 38–47, 2004.

[11] J. Gilberg. E-vote: An internet-based electronic voting system: Consolidated prototype 2 documentation, 2003.

[12] Lorrie Faith Cranor and Ron Cytron. Sensus: A security-conscious electronic polling system for the internet, 1997.

[13] Mark Allan Herschberg. Secure electronic voting over the world wide web. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1997.

[14] Aggelos Kiayias, Michael Korman, and David Walluck. An internet voting system supporting user privacy. In *ACSAC*, pages 165–174. IEEE Computer Society, 2006.

[15] Ivan Damgard, Jens Groth, and Gorm Salomonsen. The theory and implementation of an electronic voting system, January 03 2002.

[16] O. Baudron, P. a. Fouque, D. Pointcheval, and G. Poupard. Practical multi-candidate election system, 2001.

[17] Ivan Damgard, Mads Jurik, and Jesper Buus Nielsen. A generalization of paillier's public-key system with applications to electronic voting, 2003.

[18] Mihir Bellare and Phillip Rogaway. Introduction to modern cryptography.

[19] W. Diffie and M. E. Hellam. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22, 1976.

[20] Ronald John Fitzgerald Cramer. Modular design of secure yet practical cryptographic protocols, 1997.

[21] Fiat and Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO: Proceedings of Crypto*, 1986.

[22] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[23] Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1991.

[24] Gennaro, Jarecki, Krawczyk, and Rabin. Secure distributed key generation for discrete-log based cryptosystems. *JCRYPTOL: Journal of Cryptology*, 20, 2007.

[25] Fouque and Stern. Fully distributed threshold RSA under standard assumptions. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 2001.

[26] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. Advances in Cryptology – EUROCRYPT '96*, pages 387–398, 1996.

[27] Naveen Sastry, Tadayoshi Kohno, and David Wagner. Designing voting machines for verification. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, pages 22–22, Berkeley, CA, USA, 2006. USENIX Association.

[28] Orhan Cetinkaya and Ali Doganaksoy. A practical verifiable e-voting protocol for large scale elections over a network. In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 432–442. IEEE Computer Society, 2007.