DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA.

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN.

“Arquitectura de seguridad y auditoría para un sistema de voto electrónico”

ALUMNO: Josué Figueroa González.

ASESORA: Dra. Silvia Beatriz González Brambila.

RESUMEN.

El presente trabajo muestra el diseño y el desarrollo de una arquitectura de seguridad y auditoría para un sistema de voto electrónico presencial así como los resultados que se obtuvieron de su implementación. El uso de este tipo de sistemas ha ido en aumento y es necesario que generen altos niveles de confianza en los usuarios. El proyecto se enfoca en diseñar y construir una arquitectura que sirva como base para diseñar e implementar sistemas de voto electrónico seguros y auditables.

La auditoría se enfoca en la creación de elementos auditables los cuáles permiten que los resultados de una elección puedan ser verificados en caso de que se sospeche de fraude y como consecuencia hacen que los usuarios tengan confianza en el sistema. La seguridad se refiere a la creación de protocolos criptográficos que cubran las distintas etapas de un sistema de votación electrónica y que garanticen que los resultados no puedan ser modificados o si lo han sido poder detectar esas modificaciones.

La contribución de este proyecto es la creación de una arquitectura que permita resolver el problema de la plataforma segura en los sistemas de voto electrónico, establecer una serie de requisitos que los sistemas deben cumplir para ser considerados seguros y auditables, proporcionar una manera de resolver los problemas más comunes de estos sistemas y determinar un conjunto de pruebas a las que se deben someter los sistemas para verificar su seguridad y generar un mayor nivel de confianza.

ABSTRACT.

This work shows the design and development of a security and auditing architecture for an eyewitness electronic voting system and the results of implementing it. The use of these systems has been increasing and it is necessary that they generate high levels of confidence for the users. The project focuses on developing an architecture that serves as a base for the design and construction of secure and auditable electronic voting systems.

Auditing focuses on the creation of auditable elements which allow that the results of an election can be verified in case of a suspicion of fraud, and as a consequence generates greater levels of confidence in the users. Security refers to the development of cryptographic protocols that cover the different stages of an electronic voting system and guarantees that results cannot be modified or, if they have been, that these changes can be detected.

The contribution of this project is the creation of an architecture that allows to solve the problems of insecure platforms in electronic voting systems, another contribution is to establish a series of requirements that this kind of systems must fulfill in order to be considered secure and auditable, this project intends to provide a way to solve the most common problems of these systems and determine a set of tests that they must be subject to verify its levels of security and generate higher levels of confidence.

AGRADECIMIENTOS.

"Gracias a la vida, que me ha dado tanto"

A mis padres César y Lilia

Por todo el apoyo que me brindaron durante mis años de estudio y por la educación que recibí por parte de ellos.

A mis profesores y jurados del examen de grado.

Por haberme dado los conocimientos que me permitieron llegar hasta esta etapa de mi educación.

A mis familiares y amigos.

En especial a mi hermana Ivonne por estar siempre conmigo en los momentos más difíciles.

Al grupo de trabajo del proyecto "Tinochtín.

En especial a la Dra. Silvia González que incluso en los peores momentos tuvo la paciencia para continuar apoyándome y que posteriormente se convirtió en la directora de esta tesis.

ÍNDICE.

LISTA DE FIGURAS.

LISTA DE CAPTURAS.

LISTA DE TABLAS.

LISTA DE ACRÓNIMOS.

ACP	*Access Policy Control*
ACR	*Access Control Rules*
API	*Application Programming Interface*
CAST5	*Carlisle Adams y Stafford Tavares 5*
DH	*Diffie Hellman*
DRE	*Direct Recording Voting System*
DSA	*Digital Signature Algorithm*
EBI	*Electronic Ballot Image*
EVP	*Digital Envelope*
LOV	*Lectura Óptica del Voto*
OpenSSL	*Open Secure Socket Layer*
PEM	*Privacy Enhanced Mail*
RATS	*Rough Auditing Tool for Security*
RSA	*Rivest, Shamir & Adleman*
SSL	*Secure Socket Layer*
UAM-A	*Universidad Autónoma Metropolitana Azcapotzalco*
VVPAT	*Voter Verified Paper Audit Trail*

CAPÍTULO 1

INTRODUCCIÓN.

En este capítulo se presenta una breve descripción del proyecto, la justificación que muestra el problema que dio origen al mismo, los objetivos que se trabajaron, los alcances que tiene dentro de un sistema de voto electrónico, los recursos tanto de hardware como de software que se utilizaron y la metodología que se siguió durante su desarrollo.

1.1 INTRODUCCIÓN.

Los avances tecnológicos y de computación han influido notoriamente en la manera en que se realizan muchas de las actividades actuales substituyendo a las formas tradicionales. Durante años los sistemas de votación han sido llevados a cabo mediante métodos convencionales como las boletas de papel, llamadas telefónicas ó cartas y gran parte de los resultados se obtienen a través de un conteo manual. Con el paso del tiempo y el avance tecnológico surgieron los sistemas de voto electrónico, que comenzaron a ser utilizados en 1974 y eran conocidos como *DRE (Direct Recording Voting Systems)* o sistemas de almacenamiento directo del voto los cuáles originalmente eran referidos como "contadores electrónicos de votos" [Saltman, 2003].

La computación abre una puerta que puede simplificar en muchos sentidos la tarea de realizar elecciones [Arango & Sánchez, 2004], de ahí que la computadora esté sirviendo de base fundamental para la implementación de votación electrónica, la cual actualmente se divide en dos grandes grupos, presencial y remota. El voto electrónico es una realidad desde hace varios años en países como Francia, Alemania, Inglaterra, España, India, Brasil, Estados Unidos, Argentina y Venezuela entre otros, y se espera que dentro de poco este tipo de sistemas se aplique en México [FEPADE, 2004].

Sin embargo, dado el gran desconocimiento que existe sobre la tecnología con la que éstos sistemas son construidos por una parte muy grande de la población, se introduce también una gran desconfianza en muchos sentidos [Arango & Sánchez, 2004].

Las características que estos sistemas deben cumplir para que los usuarios tengan confianza en ellos son: la opción registrada debe ser la misma que ellos quisieron emitir, los resultados finales deben mostrar realmente el sentir de todos aquellos que participaron en la elección, la opción que eligieron debe ser secreta, tener la seguridad de que los votos no pueden ser alterados y sobre todo deben ser sencillos de utilizar.

Para incrementar los niveles de confianza en éste tipo de sistemas, deben estar presentes desde su diseño conceptos como el de auditoría y auditabilidad [Saltman, 2001], además deben contar con elementos de seguridad que los protejan no solo contra acciones impropias por parte de los votantes, sino que también de las que pudieran producirse por parte de los programadores, técnicos y administradores del sistema [Jones, 2004A].

1.2 JUSTIFICACIÓN.

Uno de los principales problemas en los sistemas informáticos es el de la "plataforma segura" que es cuando no se cuenta con una arquitectura bien definida para diseñar un sistema seguro, éste problema también afecta a los sistemas de voto electrónico los cuales deben contar además con una arquitectura de auditoría.

Actualmente no se cuenta con una arquitectura bien definida que ayude a diseñar de manera eficiente estos sistemas, lo que se tiene es una gran cantidad de opiniones y recomendaciones acerca de sus puntos críticos y lo que se hace es ir agregando defensas una vez que el sistema ha sido vulnerado, cuando lo correcto es que éste sea seguro y confiable desde el momento de su creación. La importancia de éste proyecto radica en identificar los aspectos vulnerables de los sistemas de voto electrónico desde el punto de vista de la seguridad y la auditoría, y de acuerdo con ello diseñar una arquitectura que permita resolverlos y construir sistemas de voto electrónico seguros y confiables.

1.3 OBJETIVOS.

El objetivo principal de éste proyecto es diseñar una arquitectura que permita construir sistemas de voto electrónico presencial seguros y auditables, a partir de éste se desprenden otros objetivos específicos como son:

- Determinar las vulnerabilidades de los sistemas de voto electrónico presencial desde el punto de vista de la seguridad y la auditoría.
- Desarrollar una metodología para resolver de manera eficiente éstas vulnerabilidades.
- Determinar los elementos necesarios para decir si un sistema de voto electrónico presencial es seguro y auditable.
- Implementar la arquitectura diseñada a un sistema de voto electrónico para determinar su eficiencia mediante el proceso de pruebas paralelas.

1.4 ALCANCES.

Un sistema de voto electrónico presencial es bastante complejo, éstos sistemas se encuentran formados por tres etapas principales, pre votación, votación y post votación. La etapa de pre votación es la encargada de la generación de los archivos que utilizará el sistema para configurarse de acuerdo al número de elecciones, candidatos que participan en cada una y datos específicos sobre el lugar en donde estará ubicado el equipo. En la etapa de votación se efectúa el proceso de recibir los votos y de generar resultados parciales que corresponden solamente al lugar en donde se encuentra ubicado el equipo. Finalmente la etapa de post votación es la encargada de la recolección de los resultados parciales de los distintos equipos, de su análisis estadístico y de la obtención de los resultados finales.

Este proyecto se enfoca en la etapa de votación que a su vez puede dividirse en las tres mismas etapas, pero con la diferencia de que aquí la etapa de pre votación se refiere a la instalación de archivos y configuración del sistema. La etapa de votación cuya entrada es la elección del votante, y su salida es su elección final considerando que se puede corregir el voto antes de confirmarlo, una vez que el votante ha completado el proceso de selección, no se permitirá gracias a la lógica del sistema poder realizar selecciones adicionales. Finalmente la etapa de post votación cuya función es realizar la suma de los votos individuales, reportar el total de votos que obtuvo cada candidato y el

número de votos nulos que se hayan obtenido cuando el votante haya elegido no votar por ningún candidato, cada una de las etapas cuenta con sus respectivos elementos de seguridad y de auditoría. Un punto que no se cubre es el que se refiere a la forma de activación del equipo, haciendo solo notar que debe asegurarse que un usuario no pueda votar más de una ocasión. La Figura 1 muestra los alcances del proyecto y sus relaciones con el resto de las etapas.

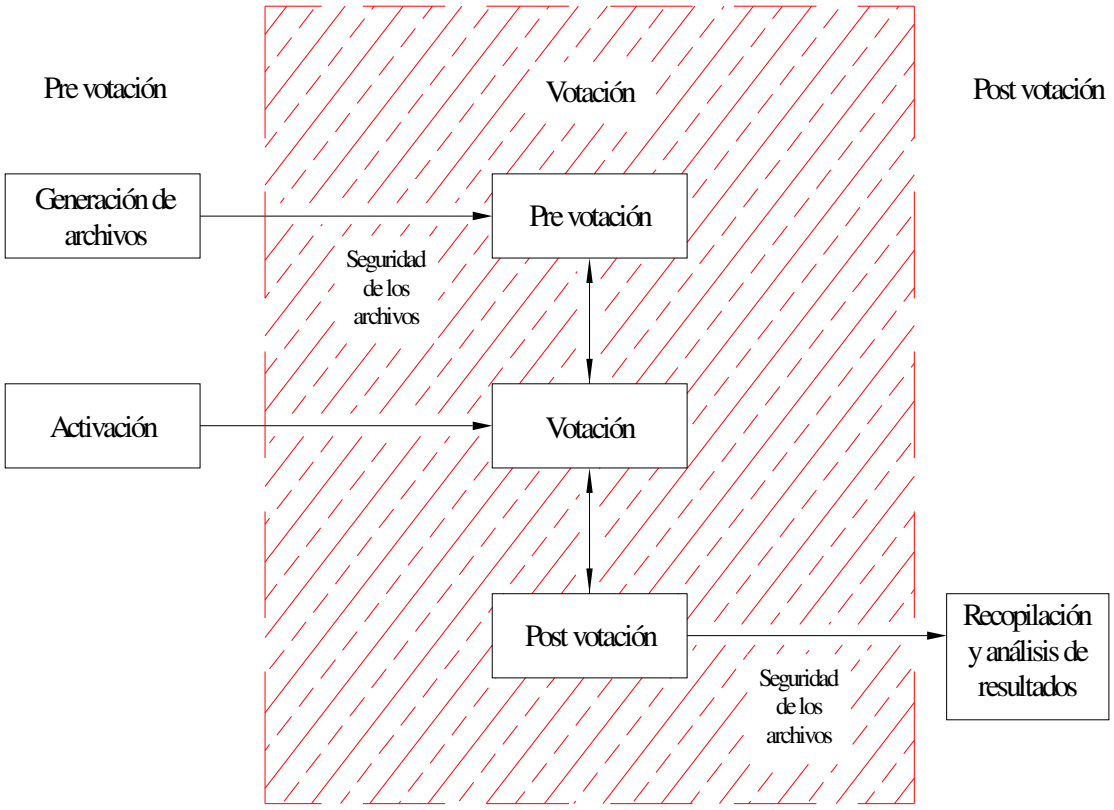


Figura 1. Alcances del proyecto.

1.5 RECURSOS.

1.5.1 Hardware.

Para el desarrollo del proyecto y sus pruebas se utilizó una computadora Pentium III a 750MHz, con 192Mbytes de memoria RAM, CD-ROM, puerto serial y puerto USB, una memoria USB para el almacenamiento de los votos, la impresión de los distintos documentos se utilizó una impresora térmica *Fujitsu* con puerto serial y para la activación del equipo se utilizó un lector de código de barras con puerto PS2.

Posteriormente el programa se implementó en una tarjeta de desarrollo *BitsyX* con procesador PXA255 32-bit *Scale* a 400MHz, 64Mbytes de memoria, memoria Compact Flash de 342MB, puerto USB, 3 puertos serial RS-32, puerto PS2, conectores de audio para audífono y micrófono y pantalla táctil LCD color con resolución 1024 x 768.

1.5.2 Software.

Sistema operativo *Linux SUSE* Profesional 8.0, bibliotecas de *OpenSSL* (*Open Secure Socket Layer*) (ver. 0.9.7), compilador para lenguaje C++ (g++) y el entorno para desarrollo de aplicaciones gráficas *QTDesigner.*

Para la implementación en un sistema incrustado se utilizó el compilador cruzado *arm-linux-g++* así como las versiones para arquitectura *ARM* de *OpenSSL*, *QTDesigner* y una versión reducida del sistema operativo *Linux Debian*.

Para realizar pruebas a la arquitectura ésta se implementa en el sistema de voto electrónico "Tinochtín" diseñado y desarrollado en la UAM-A (*Universidad Autónoma Metropolitana unidad Azcapotzalco*).

1.6 METODOLOGÍA.

El proyecto comenzó con la revisión de los antecedentes y trabajos relacionados sobre los sistemas de voto electrónico actuales, su auditoría y seguridad, también se revisaron artículos relacionados con el desarrollo de arquitecturas de seguridad y conceptos de criptografía.

Para la auditoría se seleccionaron los elementos que generaban mayor nivel de confianza en cada una de las etapas de un sistema de votación buscando que en cada una de ellas se encontrara presente el concepto de redundancia.

La seguridad consistió en diseñar un protocolo criptográfico en el que se incluyeron los distintos archivos que se generan en las diversas etapas, además del manejo que se tiene de las llaves simétricas, públicas y privadas. Para implementar el protocolo se utilizaron las distintas bibliotecas de *OpenSSL* para el manejo del cifrado simétrico y asimétrico. Para el cifrado simétrico se utilizaron las funciones que proporciona el *API* (*Application Programming Interface*) *EVP* (*Digital Envelope*) y el algoritmo de cifrado *CAST5* (*Carlisle Adams y Stafford Tavares 5*) con llaves de 128 bits de tamaño, para las funciones relacionadas con la criptografía de llave pública se manejaron las bibliotecas de *RSA* (*Rivest, Shamir & Adleman*) para la generación de llaves, cifrado, descifrado, firmado digital y verificación de firmas, el tamaño de las llaves asimétricas es de 2048 bits. El almacenamiento y la carga de las llaves asimétricas se realizaron con las funciones de la biblioteca PEM (*Privacy Enhanced Mail.*). Para la construcción de la arquitectura se tomaron los distintos niveles que menciona [Probst, 2002] y se adaptaron a las necesidades del sistema, las capas que forman la arquitectura son:

- Autenticación.
- Control de Acceso.
    - Sujeto.

  - Objeto seguro.
  - Autorización.
  - Restricciones.
- Criptografía.

Aunque [Probst, 2002] también menciona la auditoría como un elemento más de esta arquitectura, en este proyecto no se trata así, sino que cada capa cuenta además con elementos auditables.

Para combinar los elementos de seguridad y auditoría en una misma arquitectura se revisaron las distintas etapas que forman un sistema de votación electrónica y de acuerdo con esto se fueron acomodando tanto los elementos de seguridad como de auditoría en cada una de las capas que se habían seleccionado anteriormente.

La construcción de la arquitectura y su implementación en el sistema de voto electrónico se realizó de manera iterativa, lo que permitió ir generando varias versiones de ésta misma que poco a poco fueron cumpliendo con los requisitos que se habían propuesto.

La arquitectura se probó implementando sus distintos componentes en el sistema de votación electrónica "Tinochtín" desarrollado en la UAM Azcapotzalco, añadiendo o modificando los elementos necesarios, las pruebas que se realizaron al sistema se dividieron en dos, pruebas a la auditoría que están relacionadas con la confianza y pruebas a la seguridad. Para la auditoría se aplicó el esquema de pruebas paralelas [Jones, 2004B] y para la seguridad se aplicaron pruebas al protocolo criptográfico tomando en cuenta los ataques éste puede sufrir [Menezes, 1996]. Finalmente el código fuente se realizó y documentó siguiendo algunas convenciones que debe cumplir al momento de desarrollarse como son el indicar el autor, fechas de creación y modificación, nombre del archivo y de la clase. La calidad del mismo se probó con los programas *RATS* (*Rough Auditing Tool for Security*) y *Flawfinder*.

CAPÍTULO 2

TRABAJOS RELACIONADOS.

En este capítulo se revisan los trabajos de diversos autores sobre cuatro temas, el voto electrónico, su auditoría, su seguridad y temas relacionados con la arquitectura de seguridad con el objetivo de dar a conocer los conceptos y los aportes que hay sobre cada uno de éstos.

2.1 VOTO ELECTRÓNICO.

2.1.1 DEFINICIONES DEL VOTO ELECTRÓNICO.

El voto electrónico puede definirse como [Prince, 2004]:

"Aplicación de dispositivos y sistemas de tecnología de la información y telecomunicaciones al acto del sufragio total o parcialmente, a todo el proceso electoral, o a algunas de las distintas actividades del sufragio, el registro y verificación de la identidad del elector. Incluye la emisión misma del voto en una urna electrónica (con o sin impresión inmediata de boleta en papel para control del ciudadano o de la autoridad); el recuento en la mesa o el global consolidado, la transmisión de resultados, u otras actividades".

Pero el voto electrónico no es simplemente un cambio de herramientas y materiales, no significa pasar de la urna de madera o de cartón al metal y al software, es mucho más por que las posibilidades que el nuevo sistema ofrece permiten rediseñar –corrigiendo- el sistema electoral completo [Prince, 2004].

2.1.2 REQUISITOS DEL VOTO ELECTRÓNICO.

El voto electrónico al igual que el tradicional debe cumplir con una serie de requisitos, los dos más importantes son [Prince, 2004]:

- La confianza del elector en el buen funcionamiento del sistema.
- La facilidad, comodidad y sencillez que presente el sistema de emisión del voto electrónico.

Otros requisitos que un sistema de voto electrónico debe poseer son:

Anónimo y privado. Debe garantizar el anonimato y la privacidad al momento de emitir un voto. Los usuarios deben poder votar en total libertad y privacidad sin que su identidad pueda ser relacionada con su voto.

Elegible y auténtico. Solo puedan votar los usuarios autorizados, también garantizando que puedan hacerlo sólo una vez.

Íntegro. Los votos no sean cambiados o eliminados.

Exacto y verificable. Procurar el correcto almacenamiento de los votos y de toda la información que registren, y todo el proceso deberá ser verificable.

Confiable. Debe funcionar de manera robusta, sin pérdida de votos ni de datos o información. Cabe destacar que en el voto electrónico la confiabilidad se basa fundamentalmente en una cuestión de percepción por parte de los electores y no tanto en una razón técnica.

Fácil de utilizar. Debe ser fácilmente utilizable por los electores para que no genere confusiones en el elector ni en las autoridades encargadas del escrutinio.

No coaccionable. Los votantes no pueden demostrar a otros por quién votaron.

Verificable por el individuo. El votante debe poder comprobar su voto.

Un sistema de voto electrónico debe cumplir con los siguientes requisitos [Selker, 2003]:

- Asegurar el almacenamiento del voto emitido, esto se logra con una arquitectura que logre la detección y la corrección de errores.
- Prevención de alteraciones externas, especialmente las que involucren la modificación de los votos.
- Prevención de alteraciones internas que incluyen el desarrollo malicioso de sistemas de voto electrónico.
- Permitir la detección de falsificación de los contenidos de los archivos que el sistema utiliza o genera.

Para [Saltman, 2003] los sistemas de voto electrónico deben cubrir tres aspectos fundamentales:

- Debe ser sencillo para el votante emitir su voto.
- El sistema debe procesar correctamente la opción elegida.
- Debe existir confianza del público en los resultados que arrojará el sistema.

2.1.3 TIPOS DE VOTO ELECTRÓNICO.

La votación electrónica puede ser dividida en dos grandes categorías la remota y la presencial como se muestra en la Figura 2.

La votación remota se puede realizar a través de Internet mediante una PC, teléfono celular u otro dispositivo desde cualquier locación geográfica cercana o lejana al lugar donde se realizan las elecciones. La votación presencial, por su parte, implica el uso de sistemas de captación electrónica del voto, con transmisión y escrutinio provisional a través de una “urna electrónica” ubicada en los lugares físicos donde se realiza la votación tradicional [Prince, 2004].

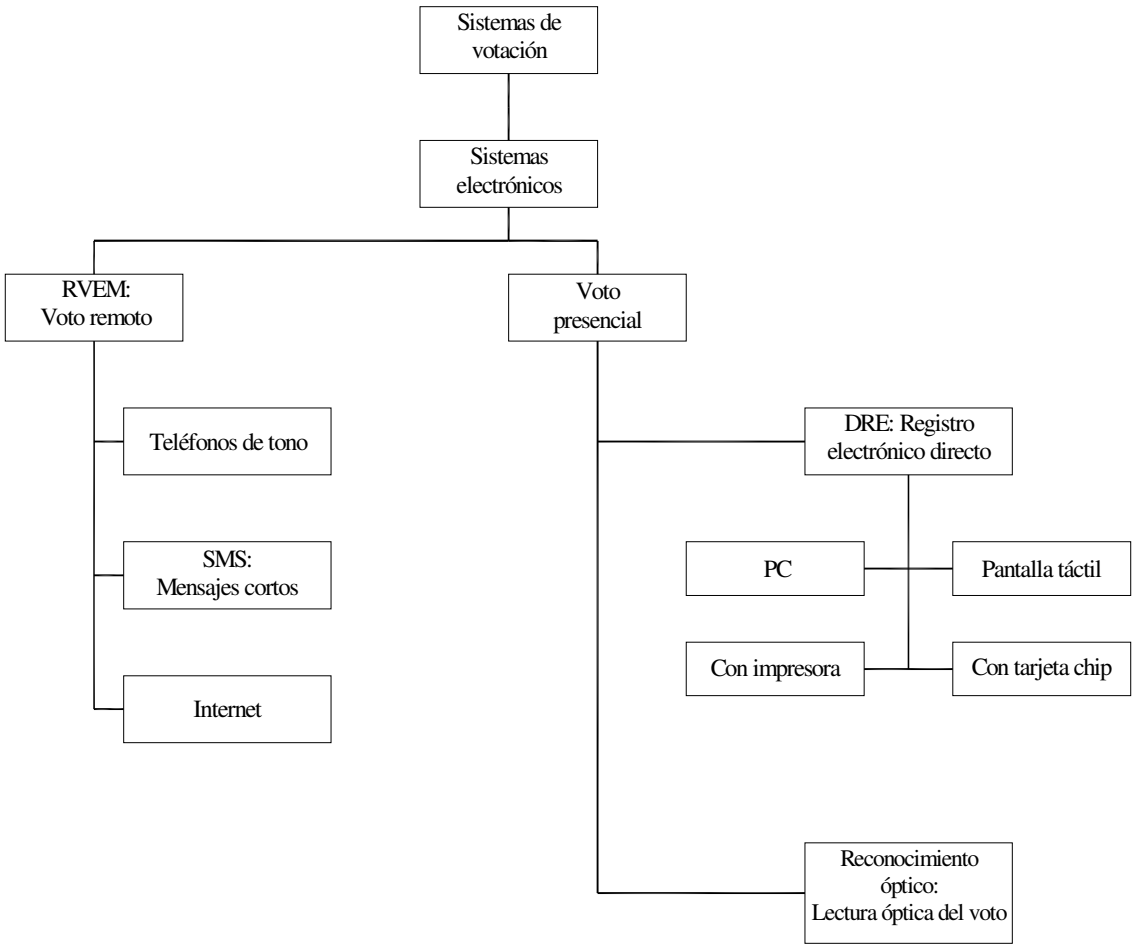


Figura 2. Tipos de voto electrónico.

Votación presencial.

La votación presencial es la que se utiliza principalmente y se pueden identificar dos grandes grupos:

- Registro Electrónico Directo (*DRE* por sus siglas en inglés).
- Lectura Óptica del Voto (*LOV*).

Estos dos se distinguen por la forma en que se emite un voto, ya sea de manera electrónica (*DRE*) o manual (*LOV*) y también por la forma en que se almacena el voto, ya sea directamente en una memoria o por digitalización óptica, pero ambos comparten una característica en común, automatizar el conteo de los sufragios y la obtención de resultados preliminares de manera casi inmediata, existen pequeñas variaciones entre ambas tecnologías las cuales pueden resumirse en la Tabla 1.

Sistema	Instrumento de votación.	Registro del voto.	Comprobante.
Sistema LOV	Boleta por elección con código de reconocimiento	Dispositivo lector óptico que identifica la boleta y registra el voto	La boleta
	Boleta Múltiple y marca manual	Dispositivo con digitalizador que lee la boleta y registra el voto	La boleta
Sistema DRE	Urna electrónica con teclado numérico	Registro del voto en la memoria del dispositivo	No utiliza la boleta, ocasionalmente se cuenta con una impresora para emitir un comprobante
	Pantalla táctil, tarjeta magnética, puntero láser	Registro del voto en la tarjeta magnética y lectura en equipo por separado	No utiliza la boleta, se usa la banda magnética de la tarjeta, ocasionalmente cuentan con una impresora para emitir un comprobante
	Pantalla táctil, tarjeta con chip	Registro del voto en la memoria del dispositivo	No utiliza la boleta, ocasionalmente se cuenta con una impresora para emitir un comprobante

Tabla 1. Diferencias entre los sistemas de voto electrónico.

2.1.4 SISTEMAS DE ALMACENAMIENTO DIRECTO DEL VOTO.

Estos sistemas llamados DRE por sus siglas en inglés, *Direct-Recording Voting Systems*, son aquellos que ya no utilizan las boletas tradicionales. Existen varios tipos, aunque podemos mencionar tres generales: los de botones, los mini interruptores y los que utilizan una pantalla táctil como interfaz.

Los de botones fueron el reemplazo del sistema de palancas que se empleaba en Estados Unidos para la elección de representantes político, en el cuál se presionan botones que contienen los logotipos de los partidos o nombres de los candidatos para emitir el voto. Los que utilizan mini interruptores son sistemas que se activan mediante el toque de los votantes sobre una superficie flexible que se coloca sobre otra en donde se muestran las opciones disponibles para elegir. El sistema más novedoso involucra el despliegue de logotipos y nombre de los candidatos en un monitor con áreas sensibles al tacto; al presionar sobre el área donde se encuentra el nombre de un candidato o logotipo, el sistema responde de acuerdo a su programación.

Secuencia en la votación presencial con un sistema *DRE*.

Desde el punto de vista del votante, la votación por medio de sistemas electrónicos no es demasiado diferente a la metodología tradicional, el procedimiento es el siguiente:

a) El elector se identifica ante las autoridades de la mesa directiva con algún documento, éstas verifican su identidad en un padrón el cuál podría ser el tradicional (en papel) o digital.

b) El elector se dirige a la máquina de votación (PC adaptada, urna electrónica, etc.) y emite su voto, el equipo puede estar en un cuarto oscuro separado, o en una cabina, tras un biombo o sistema similar que asegure el secreto.

c) La emisión se concreta tocando sobre una pantalla sensible, o eligiendo la opción por medio de un teclado. La boleta digital puede incluir fotos de los candidatos y símbolos de los partidos. En el caso de ser un elector con alguna discapacidad, los sistemas proveen alternativas como sonido, plantillas con lenguaje braille, etc.

d) Una vez hecha la selección, el sistema le permite al elector verificar las opciones elegidas antes de emitir el voto. En este momento, si lo desea, puede cambiar un número limitado de ó cuantas veces quiera sus preferencias según la configuración del programa, cuando está seguro, elige la opción “Emitir” o “Votar” y el voto se almacena en los distintos medios de almacenamiento.

e) En ciertos casos (depende de la solución) la urna puede emitir un comprobante en papel que se deposita en una urna tradicional el cual sirve para llevar a cabo auditorías.

f) El elector recibe su documento con la constancia de haber votado. Si el padrón es digital y centralizado, se asienta la emisión y éste queda inhabilitado para –si quisiera o lo intentara- volver a votar en otro sitio.

Al finalizar el horario de votación, las autoridades de mesa realizan los procedimientos para que la urna realice los conteos, emita las actas necesarias y –en algunos casos- transmita los resultados a

un centro de recopilación de datos. Esto debe hacerse del modo más seguro posible (datos cifrados, sistemas de clave pública – privada de alta seguridad; inalterabilidad de los mismos).

2.1.6 ANÁLISIS.

[Prince, 2004] da una definición de cada uno de los conceptos involucrados en el voto electrónico los cuales son muy importantes debido a la relación que guardan con los aspectos relacionados con la seguridad y la auditoría, también habla de los distintos tipos de votación electrónica, la presencial y la remota.

[Saltman, 2001] habla acerca de las etapas que conforman un sistema de voto electrónico y menciona los puntos que debe cumplir un sistema de este tipo de una manera más general, finalmente [Selker, 2003] también habla sobre los requisitos para estos sistemas de una manera también bastante general.

Algo que se puede observar en común entre los autores al analizar las propiedades de un sistema de voto electrónico es que éste debe de generar confianza en los usuarios del sistema, además de garantizar la integridad de los votos, lo cuál junto con la facilidad en su uso se pueden resumir como las propiedades más importantes que éstos sistemas deben poseer para general altos niveles de confianza hacia el público que los utiliza.

2.2 AUDITORÍA DEL VOTO ELECTRÓNICO.

2.2.1 CONCEPTOS RELACIONADOS.

En el proceso de auditoría de un sistema de voto electrónico se pueden encontrar los siguientes elementos básicos [Saltman, 2001]:

Auditoría. Realizar una auditoría de un sistema basado en el conteo es examinarlo para determinar si los resultados que reporta son adecuados de acuerdo a las entradas que recibe y a las acciones que realiza. Una auditoría sirve para corroborar que el total de votos que registró el sistema es congruente con el total de votos que recibieron en conjunto los participantes de la elección y con el total de votos que emitieron los votantes.

Auditabilidad. Es determinar cuáles de los datos en los que se basará la auditoría están disponibles para ser usados y poder obtener un reporte adecuado. Se presenta un problema en la auditabilidad cuando los datos no se encuentran disponibles o resulta muy difícil o costoso obtenerlos.

Confianza del votante. Se refiere a la certeza que tiene un votante de que su elección ha sido bien interpretada por la computadora. Además el votante quiere estar seguro de que su voto ha sido sumado correctamente a los votos que emitieron los demás usuarios.

Confianza del público en general. Nivel de aceptación que tiene el público en general, tomando como un hecho, que los resultados representan las verdaderas elecciones de los votantes. Este nivel de confianza incluye la suma de las confianzas de los votantes, además de otros factores como el desempeño de los programas computacionales, los resultados de una auditoría y los reportes de la elección.

2.2.2 AUDITORÍA DE LOS SISTEMAS DE VOTO ELECTRÓNICO.

El principal objetivo de una auditoría es detectar errores o fraudes [Jones, 2004A]. Una auditoría tiene al menos dos propósitos, el primero es determinar si la ley y los reglamentos se han seguido paso a paso en el proceso de votación y proporcionar evidencia de los errores que pudieron haber ocurrido, ya sea de manera accidental o deliberada. El segundo es contribuir al aumento de confianza en las instituciones en caso de que la auditoría demuestre que los resultados son correctos [Saltman, 2001].

Un método de auditoría es tratar de comprobar el buen funcionamiento del programa que calcula los resultados de la votación [Saltman, 2001]. Este análisis y pruebas deben ser llevados a cabo antes de que la votación comience para asegurar que los cálculos realizados por el sistema serán adecuados,

para poder probar la funcionalidad de un programa se le deben proveer votos simulados en una amplia variedad de elecciones para determinar si el programa produce un resultado pre determinado, además, el código del programa debe ser analizado para determinar si existen ciclos con código oculto que operaría solo bajo ciertas condiciones y que tendría un objetivo malicioso.

El principal problema de auditoría en los sistemas *DRE* surge debido a que no existe documentación, no hay boletas creadas por los votantes que formen registros independientes que puedan servir para un recuento parcial o total.

Es importante mencionar que una copia del voto generada por la computadora, incluso si la entrada es realizada por el votante, no constituye de manera automática un documento que sea independiente del proceso de conteo. Este documento no es creado por el votante sino que también está sujeto al buen funcionamiento del programa que lo creó. Con los sistemas *DRE* típicos el programa cuenta los votos pero la computadora puede proveer un documento que sea igual a la elección del votante, aunque puede registrar un voto que no coincida con el sentir del votante si el programa fue realizado para llevar a cabo este tipo de acciones. Una solución a esto sería que el programa imprimiera una boleta para que el votante pudiera además de verificar su voto introducirla a otro equipo o a un recipiente para fines de ser contabilizada, pero esto ya no sería un sistema de almacenamiento directo.

El diseño de un sistema *DRE* debe promover la confianza y proporcionar elementos para una adecuada revisión por parte de un auditor. La característica más deseable para contribuir en la auditabilidad y en la confianza del votante es el almacenamiento de una *EBI* (*Electronic Ballot Image*). Este archivo debe ser la salida de la sección de entrada del voto, cada *EBI* debe ser almacenada en una posición de memoria aleatoria, de tal manera que la secuencia de los votantes no corresponda a la secuencia de las *EBI*. Una vez que la *EBI* es almacenada debe convertirse al modo de solo lectura lo que asegura que una vez almacenada no podrá ser modificada. La retención de la *EBI* tiene un valor específico para permitir un recuento ya que representan el sentir de los votantes. Las *EBI* deberán ser almacenadas en un medio removible y ser insertadas en otro equipo para asegurar que se obtienen los mismos resultados.

Recomendaciones.

[Saltman, 2001] realiza una serie de recomendaciones que resumen sus ideas sobre un sistema de voto electrónico auditable.

Asegurar el buen funcionamiento de los sistemas *DRE* requiere de la implementación de ciertas características de diseño de estos mismos y de la implementación de ciertos procedimientos que aseguren el correcto funcionamiento del hardware y del software de las máquinas. Una auditoría con un alto nivel de confianza no puede realizarse sin este tipo de implementaciones.

Estos sistemas deben ser auditados asegurando el buen funcionamiento del hardware y del software, estos no pueden ser auditados sólo con el conteo de documentos generados por ellos mismos. El buen funcionamiento del software es asegurado probándolo contra un conjunto de votos simulados que contengan una amplia variedad de selecciones de candidatos y verificándolo para descubrir rutinas de código oculto.

También es recomendable que estos sistemas sean diseñados para almacenar las *EBI* de cada votante en localidades aleatorias, protegidas o en medios removibles, y que un porcentaje de estos sean recontados en equipos independientes. El porcentaje de recuento debe relacionarse con lo cerrada de la elección, entre más cerrada sea una elección más grande deberá ser el porcentaje de equipos con recuento. La retención de la *EBI* proporciona un registro de la elección de cada votante, en caso de alguna falla, también proporciona una especie de resguardo de los votos que se hayan registrado hasta ese momento.

Propuesta de un sistema auditable.

[Saltman, 2003] propone que el sistema dependa lo menos posible de los factores humanos que intervendrán al momento de la votación y con la intención de generar un nivel de confianza mayor hacia el votante y posteriormente hacia el público en general, el sistema que el propone cuenta con las siguientes características:

(1) Deshabilitación del equipo después de que se ha completado el proceso de votación: cada vez que se termine la etapa de votación para cada votante individual, el equipo se deshabilitará y no podrá recibir ningún otro voto hasta que sea habilitado nuevamente por alguno de los administradores del proceso de votación.
(2) Reporte directo hacia el votante de la contribución de su voto: una prueba que ha demostrado generar un buen nivel de confianza es un pequeño contador visible para los votantes, que se incrementa en uno después de que han votado, a pesar de que no garantiza que el voto se le otorgó al candidato de su elección, el hecho de que el número de votos se incremente en uno parece producir una mejora en el nivel de confianza.
(3) Almacenar los votos y contabilizarlos en otra máquina: es importante que los votos se almacenen en algún medio físico que pueda insertarse en otro dispositivo además de que se almacenen en la memoria interna de la urna, que servirá como respaldo de los resultados.
(4) Suma de los votos y de los votos nulos: una buena prueba para una auditoría, es la suma de los votos, tanto los votos que registró cada candidato como los votos nulos, estos pueden manejarse por separado y después sumarse, o ir incrementando algún contador cada que se vote, independientemente del tipo de voto.
(5) Asegurar que todos los tipos de elecciones se le presenten al votante: puede utilizarse un indicador que le avise al votante si le falta votar en alguna de las elecciones o el sistema puede ir presentando todas las votaciones en un orden secuencial.

(6) Facilidad de pruebas: el sistema debe ser fácil de probar por parte de las autoridades encargadas de los procesos de votación, estas pruebas deben realizarse en presencia de los participantes interesados, estas pruebas no deberán afectar el futuro desempeño del sistema.
(7) El software debe ser escrito para que sea sencillo de modificar de acuerdo al tipo de elección: se deben tener solo espacios en blanco para insertar los datos de partidos, candidatos, tipo de elección, sin tener que modificar ninguna de las rutinas de registro de votos, de totalización, o de almacenamiento.

2.2.3 LA IMPORTANCIA DE LA REDUNDANCIA.

Para [Jones, 2004A] el requisito central para que un sistema sea auditable es que contenga suficiente información para permitir una detección y corrección de un error o falsificación. En cuanto a esto, los sistemas auditables comparten muchas características con los sistemas tolerantes a fallos.

La forma básica de una auditoría.
La forma más básica pero que es muy eficiente para mostrar los resultados arrojados por una auditoría a un sistema de votación es la siguiente:

$V = C + U$

Donde:
V: Votos almacenados
C: Suma de los votos registrados por cada candidato en la elección
U: Número de votos nulos

Todas estas cantidades deben ser positivas.

Si esta igualdad no se cumple, la diferencia provee una medida del error en el conteo. Pero con este nivel de auditoría no se puede detectar el caso en que se añadan votos a un candidato y se le resten a otro, desafortunadamente esto es exactamente el resultado que se esperaría de un sistema construido cuidadosamente.

Mientras que muchos elementos auditables se desarrollan y se añaden a un sistema para propósitos de auditoría al término de una elección, su uso más importante es una auto auditoría inmediata que permita detectar errores antes de que el conteo total de los votos este completo para que éstos puedan ser detectados y corregidos antes de que los resultados sean de acceso al público en general.

Auditoría posterior a la elección. La autenticidad de los documentos puede ser realizada con técnicas criptográficas, pero un auditor necesitará técnicas más sencillas. Un aspecto importante es

la cadena de custodia de cada evento que se realizó en la elección, lo que se necesita en el caso de los sistemas de voto electrónico es un documento análogo. Estos sistemas generalmente guardan en sus archivos de bitácoras registros del momento en el que se encendió el equipo, el momento en el que se comenzó y finalizó la elección, el momento en el que se realizó cada voto pero sin descuidar el aspecto de secrecía que deben cumplir los votos.

Recuento. Si el sistema almacena las boletas originales o si se genera un registro electrónico es posible un conteo genuino. Lo que importa desde el punto de vista del auditor es que estos medios puedan ser examinados sin el uso de ninguno de los componentes de software que estén bajo el proceso de auditoría.

El asegurar que el sistema realmente ha capturado la elección del votante es difícil ya que no es tan sencillo auditar la interfaz que comunica al sistema con el votante. Una solución a este problema es presentar al votante una copia de su voto e invitarlo para que lo verifique, si se utiliza como medida sólo de auditoría y recuento, fortalece el nivel de confianza, si este documento se utiliza como una boleta o voto legal (por ejemplo, utilizándolo como entrada para ser contabilizado en otro equipo) entonces entre la auditoría realizada por los votantes y algunos recuentos ocasionales se ha conseguido el objetivo de tener una auditoría definitiva.

Las conclusiones a las que llega [Jones, 2004A] en su análisis sobre la auditoría en los sistemas de voto electrónico son:

La auditoría es más eficiente cuando los requisitos computacionales para llevarla a cabo son lo suficientemente simples para que varios observadores puedan realizar auditorias independientes. Si el entendimiento de los requisitos de auditoría se logra en una parte temprana del diseño, se puede lograr una implementación que facilite las condiciones para llevar a cabo una auditoría adecuada. Una auditoría no puede detectar todos los problemas, no puede detectar violaciones en la privacidad del votante o una incorrecta presentación de las boletas. Las pruebas, en particular las paralelas, son necesarias para protegerse de este tipo de problemas.

2.2.4 EL DOCUMENTO *VVPAT*.

[Bolin, 2005] justifica el comprobante impreso diciendo que algunos sistemas confían en su arquitectura para producir un documento auditable cuando la elección ha concluido, dándole la responsabilidad de generar confianza al mismo equipo, esto genera un problema ya que los sistemas DRE presentan dos problemas: primero, el votante no tiene una manera de saber que su voto ha sido registrado de manera correcta y segundo, al no tener ningún registro físico del voto, se anula la posibilidad de un recuento al finalizar la elección, una solución es un documento conocido como *VVPAT* (*Voter Verified Paper Audit Trail*) que contiene el sentido del voto y se presenta al votante

para que este pueda confirmar que su voto fue registrado además de servir como un medio de recuento en caso de que sea necesario.

2.2.5 EL ARCHIVO DE REGISTRO DE EVENTOS.

En su análisis sobre el voto electrónico [Prince, 2004] realiza un amplio estudio de lo que es y lo que se espera del voto electrónico, esto se puede revisar en el capítulo sobre el voto electrónico de éste trabajo, pero realiza un aporte en lo que se refiere a la auditoría detallando un poco más el contenido del archivo de eventos o bitácora.

La bitácora deberá contar con las siguientes características:
1 -Detección y registro permanente de cualquier evento significativo ocurrido con el dispositivo
2 -Existencia de un reloj, que debe ser utilizado en todos los registros de eventos
3 -El registro de eventos debe estar protegido contra fallas de energía; el dispositivo debe ser capaz de producir una versión impresa del registro de eventos

2.2.6 EL MÉTODO MERCURI.

Para [Mercuri, 2002] el escrutinio al código fuente no tiene tanta importancia, pues como mencionó Ken Thompson, un co-inventor del sistema operativo *Unix*, "No puedes confiar en un código que tu no hiciste en su totalidad", ningún nivel de verificación puede proteger a alguien de código malicioso, "un micro código bien instalado puede ser casi imposible de detectar". Esta realidad computacional afecta en extremo a los sistemas de voto electrónico.

Pruebas apropiadas al sistema pueden revelar la presencia de este tipo de problemas, la transparencia en el proceso electoral es esencial, no solo para la auditabilidad si no para incrementar la confianza del votante, esto puede lograrse de una manera muy simple a través del uso de un *VVPAT*.

En el método de [Mercuri, 2002] se resalta la importancia del comprobante impreso, aquí el sistema imprime un comprobante que contiene la elección que el votante ha realizado, éste puede revisar a través de una pantalla si realmente corresponde a su sentir y es depositado en un receptor lo que elimina la posibilidad de que pueda ser removido. Si por alguna razón, el documento no contiene la elección que el votante había emitido, un encargado del sistema que se encuentre en el lugar de la elección tendrá que retirar el papel y se le dará una nueva oportunidad al votante para que realice el proceso nuevamente.

Al final de la elección, los votos registrados por el sistema pueden servir como resultados parciales, pero los resultados oficiales de la elección vendrán de los comprobantes impresos. Dado que los comprobantes son impresos en un formato que puedan ser leídos por las personas y por las máquinas, pueden ser digitalizados o contabilizados a mano para obtener los resultados finales.

También quizá al final de la elección otra entidad podría verificar los documentos con su propio sistema, eso si la impresión se realiza de una manera que todos los sistemas de los interesados puedan leerlo.

2.2.7 AUDITORÍA DEL CÓDIGO FUENTE.

El leer el código fuente de un programa da una visión no muy completa de las acciones que éste realiza, se debe incluir comentarios detallados del funcionamiento de cada una de las distintas rutinas, funciones o módulos, de igual manera deben estar disponibles versiones anteriores del código y hacer énfasis en los cambios que se les han realizado. El código debe examinarse para evitar que incluya rutinas que alteren de manera intencionada o no el funcionamiento del programa, otra propiedad es que debe estar libre de rutinas que lo hagan caer en ciclos infinitos, que produzcan violaciones de segmentos o sobre flujos además hay que asegurar que las variables deben estar inicializadas de manera correcta.

2.2.8 PRUEBAS PARALELAS.

Para comprobar el funcionamiento del sistema y la confianza que genera en el público en general se tienen las siguientes pruebas y maneras en que pueden ser implementadas [Jones, 2004B].

El día de la elección la prueba se realiza:

- **En el lugar de la elección.** Esto ofrece el mayor realismo para una prueba y una gran oportunidad para la educación del público, desafortunadamente posee un alto riesgo de seguridad debido a que puede crear confusiones y generar desorden en el sitio de la elección.

- **En un lugar cercano al de la elección.** La presencia de un muro o una cortina entre el sitio de la elección y de pruebas reduce el potencial de distracción mientras que permite un gran realismo, desafortunadamente no todos los sitios en donde se realiza una elección tienen disponibles estos lugares.

- **En otra locación.** Esto ofrece una gran flexibilidad gran seguridad y es la que menos confusiones genera, pero reduce la participación del público y el realismo.

El sitio de las pruebas estará abierto:

- **Durante el tiempo normal que dure la elección.** Se maximiza el realismo, pero requiere que los participantes en la prueba permanezcan todo el día en el lugar donde se realiza, además que esto puede limitar los sitios donde se puede realizar.

- **Durante un horario reducido.** Esto limita seriamente el realismo y alcances de la prueba, dado que una manera de verificar que el sistema no se degrade o se corrompa es verlo

funcionando durante todo el tiempo que debería. Por otro lado hace posible que la prueba se pueda realizar en más lugares.

El equipo debe seleccionarse de manera aleatoria.

- **En el último momento antes de que comience la elección.** Minimiza la posibilidad de que el equipo seleccionado pueda haber sido modificado para la prueba, pero requiere que esta sea realizada en un lugar cercano al sitio de la elección y que además se cuenten con suficientes equipos para que no afecte el que se tome uno con fines de prueba.

- **Por adelantado.** Se maximiza la flexibilidad para realizar una prueba en cualquier lugar, pero introduce la posibilidad de que el equipo pueda haber sido preparado para engañar a las pruebas.

Los votos serán introducidos:

- **Por el público en general.** Esto maximiza la oportunidad para que el público aprenda a utilizar el sistema, pero posee un pequeño defecto en la seguridad, ya que sin ningún entrenamiento, el público podría no cooperar.

- **Por voluntarios entrenados.** Permite una participación más extensa, pero no es tan abierta como una prueba hecha por el público en general. El potencial de que las pruebas sean interrumpidas se reduce con el entrenamiento de los voluntarios.

- **Por miembros administradores.** Reduce al público al nivel de simples observadores.

Las pruebas serán:

- **Dirigidas.** Las pruebas dirigidas son poco realistas y requieren una gran cooperación de los usuarios que las realizan, este tipo de pruebas son mejor realizadas si se utilizan voluntarios entrenados.

- **No dirigidas.** Esto maximiza el realismo y la única cooperación que se necesita es una buena voluntad de los participantes para que su voto sea registrado.

Los votos registrados deben ser tabulados:

- **A mano en el lugar de la prueba:** Esto involucra una gran labor manual, pero si se considera que un número adecuado para la prueba serían 100 votos se puede tener una prueba paralela real.

- **Con una computadora conociendo los resultados.** Esto es posible solo con una prueba dirigida.

- **Con una computadora al finalizar la prueba.** Esto genera un conflicto con el concepto de tabulación y tiende a generar retrasos ya que hay que esperar algún tiempo después a que termine la prueba para conocer los resultados.

Para las salidas que el sistema genere durante la prueba:

- **Solo el comprobante impreso es comparado con la tabulación.** Esto es fácil de realizar en el lugar de la prueba con resultados ya tabulados o con un conteo manual, sin embargo el comprobante impreso es uno de los últimos elementos que se deben revisar durante una elección.

- **Los registros son leídos inmediatamente utilizando una computadora portátil.** Esto permite una prueba completa y relativamente rápida en el lugar donde se llevó a cabo la elección, pero se necesita una computadora con el software adecuado para realizar este análisis.

2.2.9 ANÁLISIS.

Al analizar el trabajo de diversos autores existen similitudes y diferencias en cuanto a las medidas de auditoría que un sistema de voto electrónico debe poseer. [Saltman, 2001] da una gran importancia a las *EBI* y al análisis del código fuente como medida para la generación de confianza. Para [Jones, 2004A] las *EBI* no son tan importantes, él se enfoca más en contar con un sistema auto auditable, con redundancia de información y con la existencia de un archivo de eventos o bitácora, el cuál [Prince, 2004] describe de una manera más detallada.

La principal diferencia que existe entre los autores es la importancia o incluso la existencia del comprobante impreso hacia el votante ó *VVPAT*, para [Saltman, 2001] éste no debe existir ya que su presencia no garantiza que el sentir del votante fue registrado adecuadamente, además de que el usar éste documento como medio para la obtención de resultados elimina la esencia de los sistemas *DRE* al recurrir de nuevo a un conteo manual, [Jones, 2004A] y [Bolin, 2005] lo trabajan como un elemento para incrementar la confianza del votante hacia el sistema mientras que [Mercuri, 2002] lo coloca como el punto fundamental en la obtención de los resultados finales. Algo en lo que los autores coinciden es en la realización de pruebas a los sistemas, las cuáles son formalizadas en el trabajo de [Jones, 2004B] al introducir el concepto de pruebas paralelas. Finalmente [Kohno et al., 2004] se enfocan en las recomendaciones a seguir al momento de crear el código fuente de un sistema de voto electrónico.

2.3 SEGURIDAD DEL VOTO ELECTRÓNICO.

2.3.1 CONCEPTOS RELACIONADOS.

A continuación se presentan algunos de los conceptos que se pueden encontrar cuando se revisa la seguridad de los sistemas de voto electrónico presencial.

Seguridad. Algunos métodos típicos de implementar la seguridad en el voto electrónico se enfocan en (1) aislar el proceso para que nadie pueda ver o modificar un voto y (2) construir el sistema bajo un esquema basado en el aislamiento como medida de seguridad, esto es conocido como "seguridad a través de la oscuridad" basada en que si nadie sabe cómo funciona el código nadie puede alterarlo, sin embargo los temas más recientes sobre seguridad hablan del valor de la revisión por parte de expertos, sobre la redundancia y sobre los modelos de código abierto.

Criptografía. Para [Fischer, 2003] el uso de la criptografía en estos sistemas proporciona un nivel más elevado de algunas de las propiedades que se cubren con la auditoría, especialmente en el aspecto de privacidad ya que además de almacenar el voto de manera aleatoria este se encuentra cifrado, lo que impide conocer su contenido y dificulta el poder modificarlo. Aunque la criptografía es sólo una pequeña parte de la seguridad de un sistema también se considera como una parte crítica que permite que algunos tengan acceso a la información y otros no [Boneh, 1999]. La criptografía no es un problema, existen una gran cantidad de algoritmos criptográficos que han sido probados satisfactoriamente, el problema principal es la arquitectura de seguridad que este tipo de sistemas deben tener, ya que según el principio de Kerckhoffs la seguridad de un sistema depende solo de la secrecía de la llave y no de la de los algoritmos [Bonhe, 1999]. Para [Selker, 2003], la tecnología existente es capaz de producir sistemas de voto electrónico seguros, confiables y auditables, estos sistemas tienen como base una arquitectura basada en la redundancia en cada una de las etapas del proceso de votación lo que los hace resistentes contra los ataques externos y contra la inserción de código malicioso.

2.3.2 AMENAZAS PARA UN SISTEMA DE VOTO ELECTRÓNICO.

Existen una amplia variedad de ataques sobre los sistemas de voto electrónico, desde los individuos involucrados en la creación, la distribución y el uso de estos, así como atacantes externos. Las siguientes amenazas deben tenerse en cuenta al momento de desarrollar un sistema de voto electrónico [Selker, 2003].

Desarrollo malicioso. Una organización, el autor del código de un sistema de votación o ambos pueden insertar código malicioso, este código puede alterar los votos, desechar algunos, o producir resultados incorrectos, además de hacer que el funcionamiento del sistema se vaya degradando.

Ataques externos. Hasta la fecha, los atacantes externos no han tenido mucho tiempo y acceso a los sistemas de voto para poder alterarlos, principalmente debido a que en el caso de los sistemas de voto electrónico presencial no se cuenta con un elemento que permita que el sistema pueda ser alterado, por ejemplo un teclado, ya que la interacción entre el equipo y el votante se limita a presionar botones o presionar sobre una pantalla táctil.

Votantes maliciosos. Un votante que obtenga un acceso indebido al sistema puede tratar de votar en más de una ocasión, votar por alguna otra persona o tratar de robar los votos de otros usuarios.

2.3.3 ARQUITECTURA DE SEGURIDAD.

Diseñar sistemas seguros requiere atención en muchos niveles, el enfoque de [Selker, 2003] comienza asegurando que no hay un solo punto para una posible falla después de que el votante ha emitido su voto.

El principio de redundancia es central, habilita al sistema para continuar trabajando incluso si ha ocurrido una falla en algún momento. Tener múltiples programas para procesar cada etapa del voto mejora la confiabilidad, sin importar quien los escribió ó cómo los escribió.

La arquitectura que él propone está compuesta de cuatro capas principales: Una interfaz con el usuario que captura los votos, el registro para asegurar que el usuario es válido, un testigo para crear registros seguros y auditables y un acumulador para producir una salida, existen otras capas que le proporcionan al votante pruebas de su voto fue registrado.

La interfaz de usuario. Quizá el componente más importante de cualquier arquitectura de votación es la interfaz de usuario, la arquitectura propuesta por [Selker, 2003] permite que los módulos de la interfaz de usuario sean desarrollados de manera independiente del resto de la arquitectura. La interfaz de usuario toma dos entradas: la definición de la interfaz y la boleta en blanco, la definición de interfaz describe la manera en que se recibe un voto, la interfaz de usuario recolecta los votos de los usuarios así como los datos de registro, después se cifra la boleta, la información de registro es añadida a los votos cifrados y los paquetes resultantes se transmiten al sistema de registro.

El sistema de registro. Es el centro de esta arquitectura de votación, el servidor tiene acceso a todos los votantes permitidos, cuando el sistema recibe un paquete que contiene información de registro y una boleta cifrada verifica si el votante es válido y posteriormente realiza una modificación al archivo de votantes para deshabilitar al votante e impedir que pueda votar nuevamente.

Cada módulo extrae la boleta cifrada, la firma y la envía al módulo testigo para obtener otras firmas, una vez que el testigo regresa las firmas estas pueden ser añadidas al dato cifrado, entonces el paquete completo (sin ningún tipo de información individual) es enviado hacia el módulo de acumulación.

El módulo testigo. Es el más sencillo de todos ya que simplemente toma una boleta cifrada y produce una firma, la boleta es firmada y se produce un valor *Hash*, el cual combinado con la llave privada del módulo testigo produce un número que hasta donde se sabe solo puede ser producido por el que tiene la llave privada.

El módulo acumulador. Este módulo toma el paquete que contiene la boleta cifrada y una serie de firmas producidas por el sistema de registro y el módulo testigo. El acumulador separa las firmas y usa la llave pública del testigo para verificarlas, luego utiliza otro conjunto de firmas para descifrar la boleta. Una vez que la boleta se encuentra en texto simple, las selecciones son almacenadas y la boleta es almacenada tanto en texto simple como cifrado.

La redundancia contra el comprobante impreso.

Una gran controversia que ha surgido últimamente es si se debe añadir un comprobante impreso en los sistemas de votación, la respuesta según [Selker, 2003] es que no debe existir, ya que es fácil de perder a comparación de un comprobante almacenado en la computadora.

Añadir comprobantes a un sistema de votación electrónica, subestima la confianza del público en el sistema. Para [Selker, 2003] en lugar de invertir dinero en crear comprobantes impresos se debería invertir en crear sistemas realmente seguros y confiables.

2.3.4 SEGURIDAD CRIPTOGRÁFICA.

Este sistema propuesto por [Selker, 2003] utiliza el siguiente esquema criptográfico para alcanzar sus objetivos de seguridad.

- Todos los módulos tienen sus propias llaves privadas.
- Los módulos firman digitalmente todo lo que transmiten, así que los datos se encuentran protegidos contra ataques intermedios al momento de la transmisión.
- Todas las transmisiones son realizadas con *SSL* (*Secure Socket Layer*), el método más confiable para transmisión que se tiene actualmente.

Además de estas medidas, se debe asegurar que el votante sea válido, el sistema de registro no debe tener conocimiento de cómo fue emitido el voto, la boleta debe estar separada del acceso al votante y contar con un cifrado que evita que el voto pueda ser observado por otros en el sistema de registro.

2.3.5 EL PROBLEMA DE LA PLATAFORMA SEGURA.

Para [Rivest, 2001], existe un problema fundamental que se debe solucionar cuando se diseñan sistemas de voto electrónico, el "problema de la plataforma segura.", la criptografía no es el problema, existen muchas técnicas que han sido probadas eficientemente, el problema es la interacción de la criptografía con los votantes. Muchos protocolos asumen que el votante tiene una plataforma computacional segura que ejecutará correctamente una parte del protocolo. El problema de la plataforma segura se presenta sobre todo en las votaciones a través de Internet, donde el sistema esta al alcance de muchas personas en la red, es por eso que debe diseñarse para evitar ataques externos.

2.3.6 SEGURIDAD DE LOS VOTOS.

Para [Cranor & Cytron, 1997] los votos son el punto fundamental de los sistemas de voto electrónico, y deben cumplir con las siguientes propiedades fundamentales:

Correctez. Un sistema es correcto si no es posible alterar un voto, si no es posible para un voto válido ser eliminado del conteo final y si no es posible contabilizar un voto no válido. En la mayoría de los sistemas con esta propiedad el resultado final será adecuado debido a que no se generan errores o si se generan pudieron ser corregidos.

Invulnerabilidad. Un sistema es invulnerable si permite que solo voten electores autorizados y que estos lo realicen solo una vez.

Privacidad. Un sistema es privado si ningún tipo de autoridad o nadie más puede hacer una relación de un voto con la persona que lo emitió y que ningún votante puede probar que votó por alguien en particular.

2.3.7 SEGURIDAD DE LOS DATOS CRÍTICOS.

[Kohno et al, 2004] además de los votos considera que existen otros datos que deben ser protegidos en los sistemas de voto electrónico, otros aspectos fundamentales en cuanto a la seguridad de un sistema de voto electrónico son:

Seguridad de los datos críticos. En los sistemas de votación, proteger la integridad y la privacidad de los datos críticos es de suma importancia, estos archivos se pueden cargar en el sistema de varias maneras, como son el introduciendo un medio de almacenamiento secundario o descargando la información de alguna red. Es importante que estos archivos tengan una forma de comprobar que provienen de donde deben y que no han sido modificados en el trayecto, para esto es importante el manejo de las firmas digitales.

Seguridad del código fuente. Al crear un sistema seguro, realizar el diseño de manera correcta es sólo un parte, ya que el diseño debe ser implementado de manera segura posteriormente [Kohno et al., 2004]. Si se tiene una implementación que se ha seguido con buenas prácticas de programación pero está algo incompleta se puede pensar que en un futuro versiones más completas del código tendrán al menos la misma calidad, aunque también puede presentarse lo opuesto, ya que es muy complicado construir un sistema seguro sobre bases inseguras. Es de vital importancia evitar las definiciones *hardcoded* en especial si se trata de información relacionada con la seguridad, por ejemplo definir dentro del código contraseñas que se utilicen para cifrar o descifrar alguno de los distintos elementos que se generen dentro del sistema.

2.3.8 ANÁLISIS.

Con la lectura de los distintos artículos sobre seguridad del voto electrónico se pueden conocer más a fondo las propiedades que deben cumplir estos sistemas, de [Cranor & Cytron, 1997] se obtienen las características de la seguridad de los votos, mientras que [Khono et al., 2004] habla además sobre la seguridad de los llamados datos críticos y del código fuente. [Selker, 2003] hace una descripción sobre la arquitectura que él propone en donde el principio básico es la redundancia, su arquitectura compuesta por cuatro capas es adecuada aunque se aplica más a los sistemas de voto electrónico remoto, pero el concepto que maneja de que cada etapa debe contar con sus propias llaves publica y privada es útil para los elementos que deben ser transportados hacia otro equipo como son los archivos críticos y de resultados.

2.4 SEGURIDAD Y ARQUITECTURA DE SEGURIDAD.

2.4.1 CONCEPTOS.

La seguridad de un sistema es una mezcla de prevención, detección y respuesta. La prevención es hacer un blanco difícil o poco atractivo de atacar, la detección involucra identificar si se realizó o se está realizando un ataque y finalmente la respuesta que permite reaccionar al ataque detectado de manera decisiva para prevenir o disminuir sus efectos [García et al., 2004].

Criptografía. Criptografía es el estudio de técnicas matemáticas relacionadas con los aspectos de la seguridad de la información tales como la confidencialidad, la integridad, y la autenticación de entidades [Menezes, 1996]. Actualmente se emplean dos tipos de criptografía, la simétrica y la asimétrica.

Criptografía Simétrica. En este tipo de criptografía se utiliza la misma contraseña o llave para cifrar y descifrar la información. Entre algunos métodos de criptografía simétrica se pueden mencionar *Blowfish*, *IDEA* (*International Data Encryption Algorithm*), *FEAL* (*Fast Data Encipherment Algorithm*), *DES* (*Data Encryption Standard*) y los más comunes que son el *3-DES*, y el *Rijndael-AES*. El usar la misma llave para cifrar y para descifrar es un problema a la hora de enviar datos, ya que el remitente debe enviar previamente la llave al destinatario para que éste pueda descifrar la información, y debe hacerlo por un canal seguro. Por lo tanto la criptografía simétrica se emplea especialmente para almacenamiento seguro de datos (solamente una persona necesita la llave). Para envío de datos es preferible la criptografía asimétrica.

Criptografía Asimétrica. Aquí se utilizan dos contraseñas o llaves, una llamada llave pública y una llamada llave privada, la información se cifra con la llave pública y se descifra con la llave privada, no presenta el problema de transmisión de la llave que tiene la criptografía simétrica ya que la llave pública no sirve para descifrar la información. Los sistemas de criptografía asimétrica incluyen el *DH* (*Diffie & Hellman*), *ElGamal*, *DSA* (*Digital Signature Algorithm*), *Merkle-Hellman*, *Chor-Rivest*, *LUC*, *McEliece*, y finalmente el *RSA* que es el más ampliamente utilizado. La criptografía asimétrica ofrece varias ventajas sobre la simétrica, como son: [García et al., 2004]

- No se requiere compartir llaves.
- Da origen al concepto de firmas digitales.
- Permite el establecimiento de identidad.

2.4.2 ARQUITECTURA DE SEGURIDAD.

Una arquitectura de seguridad es el proceso de seleccionar elementos y principios de diseño que cumplan con las necesidades de seguridad del sistema [Graff, 2003.]

Arquitectura de seguridad multicapa.

[Probst, 2002] menciona que existen varios niveles de mecanismos de seguridad como se muestra en la Figura 3. Componentes que pueden ser reutilizados para el desarrollo de aplicaciones seguras están disponibles especialmente para los niveles más bajos, llamados criptografía y comunicación segura, en los niveles más altos, como los modelos de autorización, controles de acceso, autenticación y auditoría, adecuar o realizar componentes requiere una arquitectura más específica y normalmente no se pueden realizar de manera que satisfagan los requisitos de la aplicación.

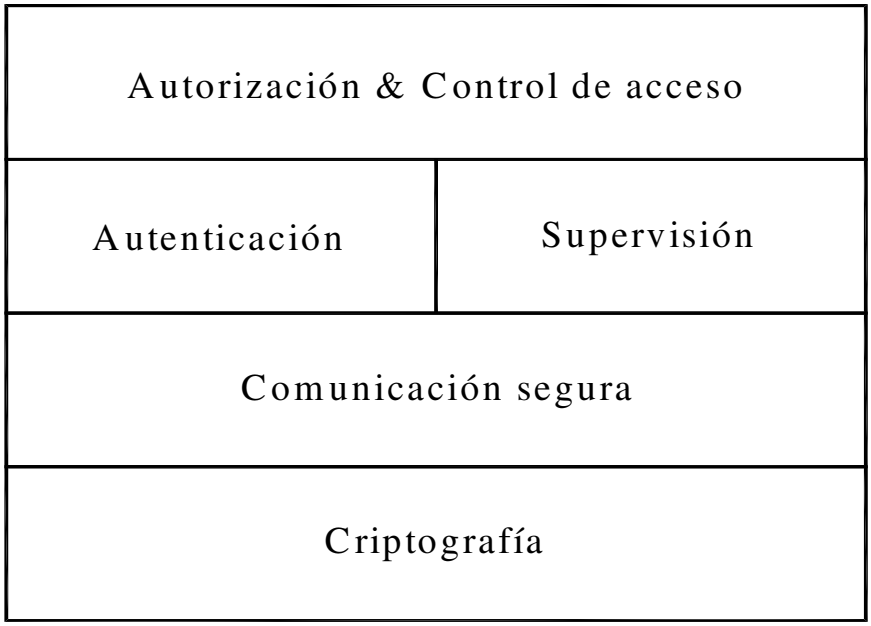


Figura 3. Niveles de un mecanismo de seguridad.

Marco de seguridad genérico.

Las aplicaciones modernas son realizadas utilizando una arquitectura multicapa, el software se divide en diversos niveles o capas de acuerdo a su funcionalidad y cada capa es capaz de comunicarse con la capa inferior o superior a ella de acuerdo a su funcionalidad a través de una interfaz bien definida.

La Figura 4 ilustra una arquitectura de capas creada para proveer mecanismos de seguridad de alto nivel en un ambiente multi-nivel.

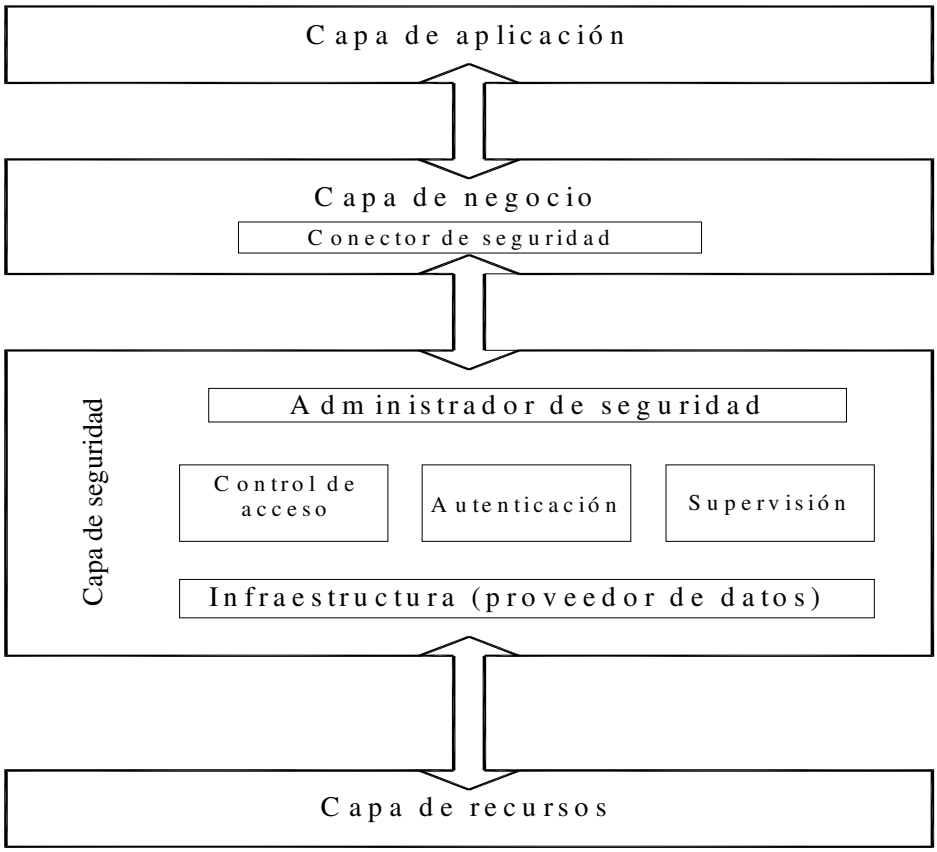


Figura 4. Arquitectura de seguridad multicapa.

En particular una capa de seguridad se establece entre la capa de negocio y las capas de recursos, ésta capa tiene una interfaz distinta a la de recursos intercambiando datos seguros con los sistemas de recursos, además tiene un componente utilizado como punto de entrada para la capa de negocio (aplicación) a la capa de seguridad, finalmente la capa de seguridad contiene componentes de seguridad de alto nivel para la coordinación en general, para proveer mecanismos de seguridad (autenticación, control de acceso, supervisión) y para la infraestructura requerida para reforzar los mecanismos de seguridad. La capa de seguridad consiste de un conjunto de clases con un objetivo en común, en este caso la seguridad y puede ser llamada marco o "*framework*".

Componentes del *framework*.

Para proveer mecanismos de seguridad de alto nivel el *framework* ofrece un coordinador central y tres componentes específicos que corresponden a la autenticación, el control de acceso y la auditoría.

Administrador de seguridad. Controla al resto de los componentes del *framework*.

Autenticación. Responsable de asegurar la correcta autenticación en la que los futuros controles de acceso estarán basados, éste verifica la identidad del sujeto basado en un identificador. El *framework* no requiere un tipo especial de identificador, por lo que el desarrollador del software puede utilizar varios mecanismos de autenticación, es tarea del componente de autenticación validar al identificador de acuerdo con el método que se utilice. El *framework* puede ofrecer un método basado en contraseñas.

Controlador de acceso. Es responsable de controlar el acceso a los objetos de acuerdo con un modelo de control de acceso particular y basado en una autenticación válida.

Supervisión. Debe ser capaz de registrar actividades de seguridad relevantes, el *framework* provee un método flexible de auditoría que recolecta mensajes del resto de los componentes y opcionalmente los puede almacenar en diversos medios. El sistema de auditoría permite filtrar los mensajes para cada medio de salida lo cual puede ser utilizado para imprimir mensajes críticos directo en la pantalla y otro tipo de información en una base de datos.

Componentes del control de acceso.
A continuación se describen los componentes relacionados con el control de acceso.

Sujeto. Se refiere a los actores o entidades del sistema, como personas, procesos o entidades.

Objeto seguro. Es la base para todas las clases de objetos que necesitan ser protegidos dentro del *framework*. Para asegurar esto es necesario que el cliente no obtenga una referencia directa al objeto.

Autorizaciones. Contiene el tipo de derechos de acceso a un recurso. El *framework* ofrece componentes especializado para una autorización positiva (permisos) o una autorización negativa (prohibiciones).

Restricciones. Permite autorización más restringida dentro del sistema en un modo más flexible, ejemplos de estas restricciones incluyen la localización (por ejemplo accesos desde una dirección IP específica) o restricciones de tiempo (accesos permitidos solo entre determinadas horas).

Modelo del control de acceso. Este componente recolecta sujetos, objetos, autorizaciones y restricciones de los componentes proveedores de datos y los transfiere a la base de autorización.

Cuando el controlador de acceso contacta a éste modelo para manejar una petición se genera una búsqueda que contiene al sujeto que realizó la petición y al objeto que esta siendo requerido,

después manda este patrón al modelo de autorización donde las reglas son buscadas y analizadas, dependiendo de las reglas uno de los siguientes resultados es regresado al controlador de acceso:

Verdadero: Si una regla es encontrada y el acceso es concedido, por ejemplo el sujeto tiene un permiso para acceder al objeto.

Falso: Si una regla es encontrada y el acceso es denegado, por ejemplo una prohibición niega el acceso del sujeto al objeto.

Mecanismo de control de acceso canónico.

Para proveer controles de acceso genéricos se debe encontrar una manera flexible de reforzar los controles de acceso. La premisa básica es que los controles de acceso pueden ser establecidos en términos de los sujetos que acceden a los objetos. La Figura 5 ilustra los pasos particulares para un control de acceso canónico.

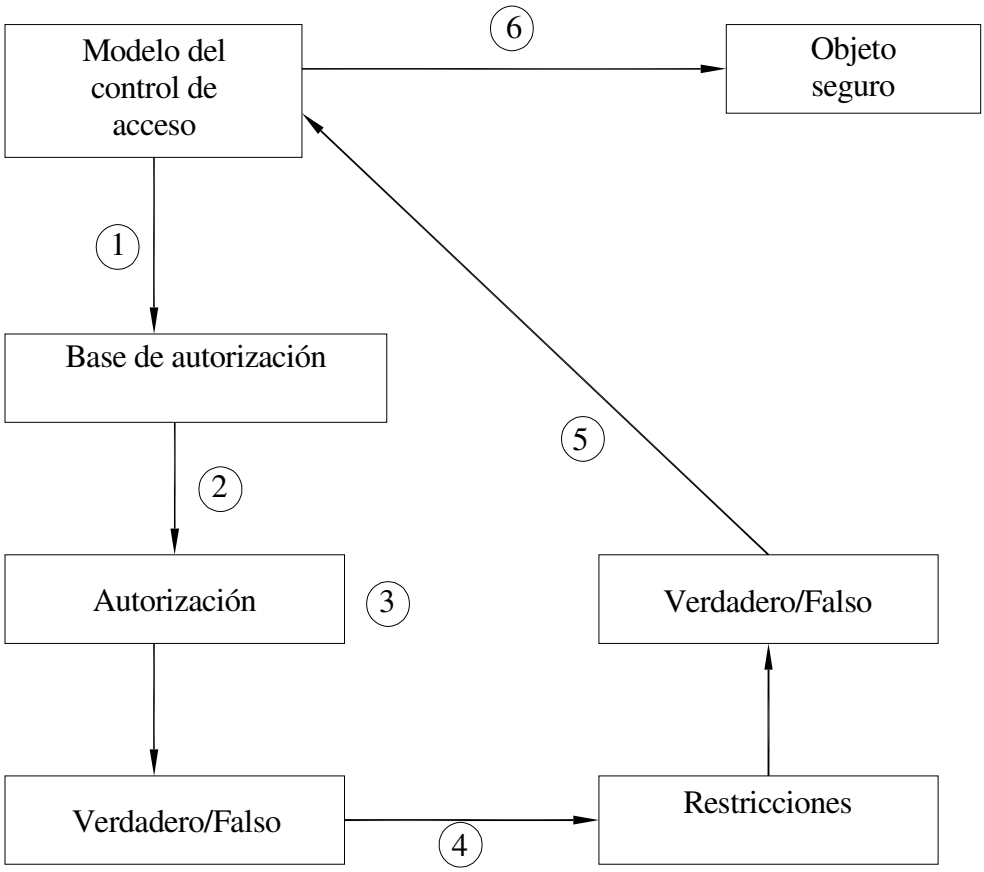


Figura 5. Pasos en un control de acceso canónico.

En general, un sujeto desea acceder a un objeto protegido de alguna manera, la operación solicitada en el objeto define las autorizaciones que son necesarias para realizar esta tarea. La manera en que se decide si el acceso es aprobado o no es determinada por el modelo de control de acceso, las autorizaciones definidas por ese modelo, y las restricciones asignadas a esas autorizaciones.

Primero el controlador de acceso recibe una petición de un sujeto ya identificado para cierta operación sobre un objeto protegido, cada modelo de control de acceso busca una combinación sujeto/objeto en la base de autorización (1) que concuerde con la petición. El proceso de búsqueda regresa una lista de reglas de autorizaciones que concuerdan con la combinación sujeto/objeto. Cada autorización es explícitamente verificada invocando un método de verificación (2). Sin embargo, existe la posibilidad de definir restricciones adicionales que restrinjan un poco más un acceso a un objeto protegido, nuevamente se verifican estas restricciones adicionales (3/4). Cuando ambas, la autorización y la restricción conceden el acceso, el modelo de control de acceso reporta un resultado positivo (acceso concedido) o negativo (acceso denegado) dependiendo del modelo (5). Este resultado es entonces regresado al controlador de seguridad quien finalmente da o niega el acceso al objeto solicitado (6).

2.4.3 ASPECTOS DE SEGURIDAD EN UN SISTEMA DISTRIBUIDO.

[Bidan, 1997] realiza un análisis de los elementos que se pueden encontrar en una arquitectura de seguridad.

Para él, la seguridad significa protección en contra de accesos no autorizados a la información, está relacionada con la confidencialidad (la información sólo es accesible a los usuarios autorizados a ella), la integridad (la información puede ser sólo modificada por usuarios que tengan el permiso para hacerlo) y la disponibilidad (el uso del sistema no puede ser negado de una manera maliciosa). La seguridad es reforzada utilizando funcionalidades de seguridad como el cifrado, la autenticación y el control de acceso.

El cifrado consiste en hacer la información ilegible (para asegurar la confidencialidad), inalterable (firma digital para asegurar la integridad) o ambas, el cifrado se utiliza para proteger información almacenada o para el intercambio de información contra lectura o modificación.

La autenticación permite asegurar la identidad de la entidad, esto es, verificar que la entidad (un usuario o proceso) es quien asegura ser. Más específicamente, el protocolo de autenticidad permite asociar cada operación del sistema con un usuario único, permitiendo verificar si la operación está permitida o no, un ejemplo de un protocolo de autenticidad es cuando se solicita un nombre de usuario y una contraseña.

El control de acceso rige las operaciones en las entidades del sistema. Por ejemplo, el control de acceso a un sistema de archivos consiste en verificar cuáles usuarios están autorizados a acceder a los archivos, el control de acceso también verifica la interacción entre entidades, además trata con el flujo de información entre entidades. El control de acceso depende de la autenticación, la identidad de la entidad debe ser única e inolvidable.

Especificaciones del cifrado.

Los datos son calificados como texto simple (*plaintext*) cuando el acceso a estos permite conocer la información que contienen, el proceso de transformar esos datos de una manera que esconda la información contenida es cifrar la información, el dato resultado es conocido como texto cifrado (*ciphertext*), el proceso de convertir un texto cifrado en texto simple es conocido como descifrado. Un algoritmo de cifrado está compuesto por una función de cifrado y su correspondiente función de descifrado, la función de cifrado debe asegurar ya sea la confidencialidad o la integridad.

En general, las funciones de cifrado y descifrado no son secretas ya que la seguridad del algoritmo de cifrado está basado en el uso de las llaves. La función de cifrado toma como entrada el texto simple y una llave de cifrado para calcular el texto cifrado, de manera inversa, dado un texto cifrado, el texto simple es calculado utilizando la correspondientes llave y función de descifrado como se muestra en la Figura 6.

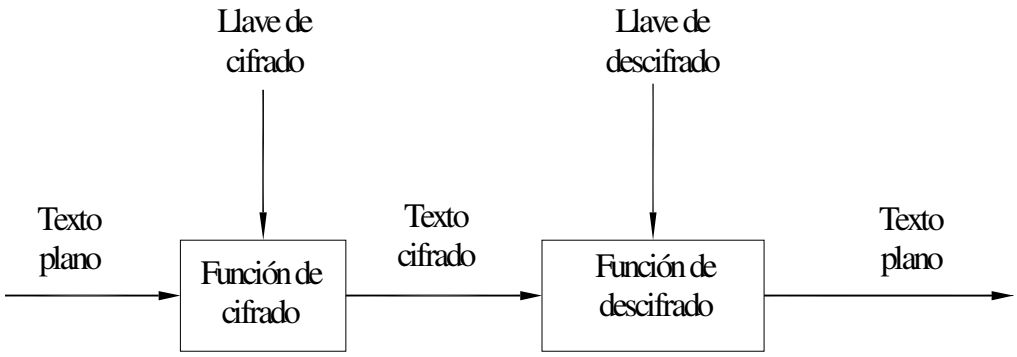


Figura 6. Esquema de cifrado y descifrado.

Especificación de los requerimientos de cifrado.

Se basa en especificar el algoritmo que se va a utilizar, así como los parámetros que describen su comportamiento. Los parámetros deben incluir: uso del algoritmo, esto es, si se utiliza para cifrar los datos (asegurar confidencialidad), firmarlos (asegurar la autenticidad), realizar una función *Hash* (para detectar manipulación) y el manejo de las llaves que incluye el tamaño, forma en que se almacenan y durante cuanto tiempo serán útiles.

Especificaciones de la autenticación.

Las entidades de autenticación permiten verificar que las entidades sean quienes aseguran ser. Un protocolo de autenticación especifica el proceso de autenticación, esto es, las entidades que participan, el intercambio de mensajes entre estas entidades y el formato de estos mensajes (texto simple o cifrado), sin embargo, el principal objetivo de estos es la autenticación de las entidades, algunos de ellos deben además tratar con propiedades adicionales de seguridad.

Especificaciones del control de acceso.

Una *ACP* (*Access Policy Control*) define el conjunto de reglas llamadas *ACR* (*Access Control Rules*) que se especifican para una pareja de entidades (ε_1,ε_2) que pueden ser usuarios, procesos, archivos, etc., un ejemplo es una regla que especifique si la entidad ε_1 tiene permitido el acceso a ε_2 o no.

2.4.4 PROTOCOLOS CRITPOGRÁFICOS.

Para [Menezes, 1996] los diferentes elementos que se encuentran presentes y que hay que considerar cuando se trabaja con elementos criptográficos son:

Primitivas criptográficas. Son las herramientas básicas que se utilizan para proveer seguridad, estas incluyen los algoritmos del cifrado simétrico, los del cifrado asimétrico y algunas que no requieren de llaves como las funciones *Hash*, o los elementos aleatorios.

Protocolo criptográfico. Es un algoritmo distribuido definido por una secuencia de pasos que especifican las acciones requeridas por dos o más entidades para lograr un objetivo específico de seguridad.

Mecanismo criptográfico. Término más general que abarca protocolos, algoritmos y técnicas no criptográficas para lograr sus objetivos de seguridad.

Falla de protocolo. Ocurre cuando un mecanismo falla en alcanzar los objetivos para los que fue creado, de una manera en que un adversario gana ventaja, no rompiendo alguna primitiva como un algoritmo de cifrado directamente, pero si manipulando el protocolo.

Algunas causas de la falla de un protocolo son:

- Debilidad en alguna de las primitivas que pueda ser amplificada por un mecanismo o un protocolo.
- Medidas de seguridad que son subestimadas o no son entendidas claramente.
- Sobrestimar algunos principios de seguridad aplicables a una amplia clase de primitivas como el cifrado.

2.4.5 ATAQUES A LOS ESQUEMAS DE CIFRADO.

El objetivo de este tipo de ataques es recuperar el texto simple a partir del texto cifrado, o peor aún deducir la llave de cifrado, algunos de los tipos de ataques que se pueden encontrar son:

Ataque a texto cifrado. Es cuando un adversario trata de de deducir la llave de cifrado o el texto simple observando el texto cifrado, cualquier esquema de seguridad vulnerable a este tipo de ataque es considerado absolutamente inseguro.

Ataque a texto simple elegido. Es cuando se elige un texto simple y se obtiene su correspondiente texto cifrado, posteriormente el adversario usa esta información para deducir el texto simple que corresponde al texto cifrado que contiene la información que desea conocer.

Ataque adaptativo. Se realiza a un texto simple elegido en donde la elección del texto simple dependerá del texto cifrado recibido.

Ataque a un texto cifrado elegido. Es cuando el adversario selecciona el texto cifrado y luego obtiene el correspondiente texto simple.

2.4.6 ATAQUES A LOS PROTOCOLOS.

Al paso de los años, se han identificado diferentes tipos de ataques a protocolos criptográficos, los ataques que se pueden realizar se clasifican en ataques pasivos y activos. Un ataque pasivo es en el que el adversario solo se desea conocer la información enviada por el canal de comunicación, un atacante pasivo solo se enfoca en la confidencialidad de los datos. Un ataque activo es cuando el adversario trata de borrar, añadir o alterar la información transmitida, un atacante activo se enfoca en la integridad, la autenticación y la confidencialidad de los datos. Para [Menezes, 1996] algunos de los ataques que se llevan acabo sobre los protocolos son:

1. Ataque de llaves conocidas, es cuando el adversario tiene acceso a llaves utilizadas anteriormente y con ellas puede determinar llaves nuevas.

2. Ataque de repetición, es cuando el adversario registra una conversación y la repite posteriormente, ya sea entera o solo una porción.

3. Personificación, es cuando el adversario asume la identidad de uno de los participantes legítimos.

4. Ataque de diccionario, se realiza comúnmente contra las contraseñas, típicamente una contraseña se almacena como una imagen de una función *Hash*, aquí el adversario tiene una lista de posibles contraseñas y trata de comparar los *Hash* de estas con el de las contraseñas almacenadas.

5. Búsqueda, es un ataque similar al de diccionario pero aquí lo que se trata de descifrar es el mensaje.

2.4.7 MODELOS PARA EVALUAR LA SEGURIDAD.

La seguridad de los protocolos y de las primitivas criptográficas puede ser evaluada bajo diferentes modelos. Los más prácticos son el computacional, el probable y el *ad hoc*.

Seguridad probable. Un método criptográfico es *probablemente seguro* si la dificultad para vencerlo puede mostrarse como igual de difícil que resolver un problema que ya se sabe que es complicado (típicamente teoría de números), como la factorización de enteros o el cálculo de logaritmos discretos.

Seguridad computacional. Esta mide la cantidad de esfuerzo computacional requerido utilizando los mejores métodos conocidos actualmente para vulnerar un sistema, se debe asumir que los sistemas deben haber sido bien estudiados para determinar qué ataques son relevantes. Una técnica se dice que es computacional mente segura si el nivel computacional requerido para vulnerarla supera por un margen considerable los recursos computacionales del adversario.

Seguridad *ad hoc*. Esta aproximación consiste de una variedad de argumentos que son mayores que los que posee el adversario. Las primitivas criptográficas y los protocolos que pasan a este análisis se dice que tienen una seguridad heurística.

2.4.8 DISEÑO DE UNA ARQUITECTURA.

Para realizar el diseño de una arquitectura de software es importante considerar el ciclo de vida del desarrollo de software, que comienza con la identificación de una necesidad y termina con la verificación formal del software desarrollado en contra de esta misma [IPL, 1997].

Para el desarrollo de una arquitectura se tienen varios métodos, entre los que se encuentran el secuencial o de cascada y el progresivo o iterativo. El proceso secuencial se basa en los siguientes pasos: recopilación de requisitos, desarrollo, pruebas y entrega final. El método progresivo consta de las siguientes partes: requisitos, diseño, implementación y pruebas y revisión, pero a diferencia del secuencial, aquí se tiene un ciclo iterativo que permite ir creando versiones del sistema que si bien al principio no cumple con los requisitos establecidos si cuenta con una funcionalidad para realizar distintas pruebas y con base en ello modificar el diseño o la forma de implementación, una vez que las pruebas son satisfactorias esa versión sirve como base para aplicar nuevamente los pasos del proceso iterativo y así hasta que el sistema finalmente cumple con los requisitos deseados.

2.4.9 ANÁLISIS.

La arquitectura multicapa que presenta [Probst, 2002] resulta adecuada para el desarrollo del proyecto con algunas modificaciones, pero los elementos que forman parte de la capa de seguridad como la autenticación, el control de acceso y la criptografía son elementos que deben estar presentes.

Por otra parte el modelo de control de acceso canónico resulta útil para el proyecto ya que su funcionamiento es muy similar al que tiene un sistema de voto electrónico si tomamos en cuenta que igual cuenta con los cuatro elementos que se describen, el sujeto, el objeto protegido, la autorización y la restricción.

La principal aportación de [Menezes, 1996] es el concepto de protocolos criptográficos, ya que la seguridad como se ha mencionado no se basa en los algoritmos de cifrado sino en el manejo de las llaves, además habla sobre los distintos tipos de ataques que pueden sufrir tanto los protocolos como los datos cifrados y eso es de utilidad al momento de diseñar el conjunto de pruebas que se aplican al sistema.

CAPÍTULO 3

DESARROLLO.

En este capítulo se explica el modelo de votación que se utilizó durante el desarrollo del proyecto, se describe el desarrollo del trabajo que incluye los elementos de auditoría que se eligieron para generar un mayor nivel de confianza, el desarrollo de la seguridad que describe el protocolo criptográfico que se diseñó y la manera en que se manejan los distintos tipos de llaves y de firmas, finalmente en la parte de desarrollo de la arquitectura se muestran las diferentes capas que la conforman, la manera en que se acomodaron los elementos de auditoría y seguridad en cada capa y la forma en que interactúan.

3.1 MODELO DE VOTACIÓN.

El modelo de votación que se siguió para el desarrollo de los elementos de auditoría y de seguridad está basado en tres etapas cómo se ha revisado en el capítulo de trabajos relacionados. La primera es la de pre votación que inicia con el encendido del sistema, siguiendo la verificación de los componentes del equipo, posteriormente aparece la parte de verificación de la integridad y autenticidad de los datos de configuración, a continuación el equipo se configura y se termina con la impresión de los distintos documentos generados. La etapa de votación inicia con la habilitación del equipo para que un usuario pueda emitir su voto, éste tiene la opción de corregir o confirmar su elección y una vez que termina de participar en todas las elecciones el sistema se deshabilita. La etapa de post votación comienza con la habilitación del sistema por un administrador, aquí se realiza el conteo total de los votos, la generación del archivo de resultados y la impresión de las actas con los resultados finales.

3.2 DESARROLLO DE LOS ELEMENTOS DE AUDITORíA.

Para realizar un sistema auditable se tomó como elemento básico la redundancia, por lo que se tenían varias maneras de presentar y almacenar la misma información, los elementos que se consideran que deben estar presentes en cada una de las etapas del proceso de votación para crear un sistema auditable son los siguientes:

3.2.1 ETAPA DE PRE VOTACIÓN.

Los elementos que se desarrollaron para esta etapa fueron:

Comprobante del funcionamiento de componentes. Los datos del procesador se obtuvieron del archivo */proc/cpuinfo* que se genera cada que inicia el sistema operativo *Linux*, los datos que se obtenían eran el tipo de procesador y la frecuencia a la que trabaja. La cantidad de memoria del equipo se obtuvo del archivo */proc/meminfo* que también se genera cada que arranca el sistema operativo, para la prueba de impresión primero se configuraba el puerto serial */dev/ttyS0* y se enviaba una cadena para que fuera impresa pidiéndole al usuario que confirmara si se había impreso de manera correcta, del video se verificaba la resolución utilizando una función del ambiente gráfico *QTDesigner*, para el sonido se pedía al usuario que se colocara los audífonos y escuchara un sonido que se reproducía haciendo uso de las funciones de *QTDesigner* para manejo de archivos de sonido (en formato *WAV*) pidiéndole que confirmara si escuchaba el sonido, para los medios de almacenamiento, solo se creaba un archivo en cada uno de los medios removibles, se trabajó con los medios representados por los directorios */mnt/sda/* y */mnt/hda/*.

Datos de configuración. Se solicita confirmar si los datos de configuración que se mostraban en pantalla eran los correctos, si lo eran se procedía al siguiente paso, de lo contrario se reportaba el error en la bitácora y el sistema se apagaba.

Actas iniciales. Aquí se generan las actas de apertura con los datos específicos del lugar donde se encuentra el equipo, posteriormente se generan las actas con los datos de cada elección y el número de votos de cada contendiente, al ser las actas iniciales, estas deben tener cero votos para cada participante, la cantidad de votos se muestra en número y letra.

3.2.2 ETAPA DE VOTACIÓN.

Para esta etapa se desarrollaron los siguientes elementos:

Comprobante impreso. Formado por el nombre de la elección y la opción elegida.

Comprobante electrónico. Formado por los números de elección y opción elegida separados por el caracter "-".

Reporte directo hacia el votante. Se muestra en la pantalla de bienvenida el número de votos que se ha registrado en cada elección, al finalizar la participación del usuario, en la pantalla de agradecimiento se muestra nuevamente el número de votos que debió haberse incrementado en uno.

Auditoría durante la elección. Cada determinado número de votos se genera un documento con los resultados hasta ese momento, una vez que se tiene más de uno de estos archivos, se verifica el más reciente con el inmediato anterior, revisando que ningún candidato tenga menos votos en el archivo más reciente que los que tenía en el conteo anterior, si esto ocurre se registra el error, éste procedimiento también se realiza para el total de votos registrados en cada elección.

Almacenamiento en diversos medios. Para que el sistema sea considerado robusto, en todo momento los votos registrados deben ser los mismos en cada uno de los medios de almacenamiento, que son una memoria USB, una memoria Compact Flash (CF) y el disco duro del equipo. Se debe almacenar en un medio una vez que se aseguró que ya se ha registrado el voto en otro de los medios.

3.2.3 ETAPA DE POST VOTACIÓN

- Para la etapa de generación de resultados, los elementos generados fueron los siguientes:

Actas finales. Una vez que termina la elección, se generan las actas finales con los datos de la elección y los resultados, éstas incluyen el nombre de cada contendiente y el total de votos en número y letra que han recibido, éste documento también se obtiene de forma impresa.

Archivo de resultados finales. Este archivo contiene los resultados finales en un formato específico que facilite su análisis posterior, se incluyen todos los tipos de elecciones y participantes de cada elección, incluyendo los votos nulos así como un número que indica el total de votos registrados en cada elección.

Pantalla de resultados finales. Se muestran los resultados finales de cada elección para poder realizar una comparación inmediata con la información contenida en las actas.

Acta de cierre de la votación. Este documento es similar al acta de apertura, pero en este caso contiene la hora en que se da por terminado el proceso de votación.

De ésta manera se cuenta con tres elementos para un recuento en caso de duda sobre los resultados arrojados por las actas finales, se cuenta con los comprobantes impresos, los comprobantes electrónicos (*EBI*) y el archivo de votos almacenado en diversos medios, y para verificar el total de votos en cada una de las elecciones se cuenta además con la información contenida en la bitácora.

El archivo de registros o bitácora. Es un elemento especial de la auditoría ya que se encuentra presente en todas las etapas del proceso de votación, se forma con la hora y la clave del evento que ocurrió, la Tabla 2 muestra los eventos y las claves con las que se registran.

Evento ocurrido	Clave del evento
Fecha de encendido incorrecta	100
Apagado	101
Fecha de encendido correcta	102
Encendido	103
Verificación de componentes	104
Audio correcto	105
Audio incorrecto	106
Configuración correcta	107
Impresión correcta de documentos iniciales	108
Impresión incorrecta de documentos iniciales	109
Configuración incorrecta	110
Configuración del equipo	111
Habilitación válida	112
Habilitación inválida	113
Finalizó el voto correctamente	114
Impresión correcta del voto	115
Impresión incorrecta del voto	116

Entrada al modulo de administración	117
Finaliza la jornada electoral	118
Impresión incorrecta de documentos finales	119
Impresión incorrecta de documentos finales	120
Deshabilitar equipo	121
Memoria USB correcta	122
Memoria USB incorrecta	123
Memoria Flash correcta	124
Memoria Flash incorrecta	125
Falló impresora	126
Auditoría intermedia correcta	127
Auditoría intermedia incorrecta	128

Tabla 2. Eventos registrados en la bitácora

La emisión de un voto también se registraba, pero solamente la elección en la que se había participado con el fin de no violar la propiedad de confidencialidad. El registro era de la siguiente manera:

NUMERO DE LA ELECCIÓN * 10 + 1 (cuando se corrige la elección) ó **+ 0** (cuando se confirma)

La Tabla 3 contiene los elementos generados, la etapa en la que aparecen y el formato en el que se presentan.

Elemento	Etapa	Formato
Funcionamiento de los componentes	Pre votación	Impreso / pantalla / archivo
Datos de configuración	Pre votación	Impreso / pantalla / archivo
Actas iniciales	Pre votación	Impreso / archivo
Comprobante impreso	Votación	Impreso
Comprobante electrónico (*EBI*)	Votación	Archivo
Reporte directo hacia el votante	Votación	Pantalla
Resultados de auditorías intermedias	Votación	Archivo
Almacenamiento en diversos medios	Votación	Archivo
Actas finales	Post votación	Impreso / archivo
Archivo de resultados finales	Post votación	Archivo
Acta de cierre de la votación.	Post votación	Impreso / archivo
Pantalla de resultados finales	Post votación	Pantalla

Tabla 3. Elementos auditables generados en las distintas etapas.

3.3 DESARROLLO DE LOS ELEMENTOS DE SEGURIDAD.

Para el manejo de la seguridad se consideraron las tres etapas que forman el proceso de votación completo, generación de medios, votación y análisis de resultados. Para realizar un protocolo criptográfico lo primero que se debe revisar es qué datos se desean proteger y qué tan importante es su confidencialidad, asegurar que nadie conozca su contenido, y su integridad, protegerlos de una posible modificación o si han sido modificados poder detectarlo. Antes de analizar el desarrollo del protocolo criptográfico es importante conocer la siguiente nomenclatura:

Generador de medios.

Llave pública cifrada	$\hat{}e_{GM}$
Llave privada	d_{GM}
Llave simétrica	k_{GM}
Llave especial	k_{ESP}

Urna electrónica.

Llave pública	e_U
Llave privada	d_U
Llave simétrica	k_U
Llave privada para firmar	d_{Uf}
Llave pública para verificar la firma	e_{Uf}

Análisis de resultados.

Llave pública	e_R
Llave privada	d_R
Llave simétrica	k_R
Llave especial	k_{ESP}

Generales.

Datos	a
Función Hash	H
Archivo cifrado	c
Firma digital	s
Firma digital de varios archivos	s_i
Password	p
Resultados	r
Resultados tecleados	r_T
Dato modificado	*

3.3.1 MANEJO INICIAL DE LAS LLAVES.

El manejo inicial de las llaves pública y privada es fundamental para el buen funcionamiento del protocolo, la generación de éstas debe realizarse en diferentes equipos, uno que genere llaves para el equipo generador de medios, otro para la urna electrónica y uno más para el equipo que analiza los resultados, comunicándose entre ellos para intercambiar las llaves necesarias, con esto se evita tener todas las llaves en un solo equipo. El instalar las llaves con anterioridad en los equipos permite defenderse del ataque más obvio que es el de suplantación, que consiste en crear un juego propio de llaves y que el sistema al cargarlos de algún dispositivo las tome como válidas.

La Tabla 4 muestra la ubicación de las distintas llaves en los diferentes equipos para poder implementar este protocolo. Las llaves simétricas se generan en el momento que se requieren y son las que se transportan por un canal de comunicación que se considera inseguro, de estas llaves no se requiere que esté instalada ninguna en alguno de los equipos.

Generador de medios	Urna electrónica	Equipo de resultados
Llave publica de la urna electrónica (e_U) Llave privada del generador de medios (d_{GM})	Llave privada de la urna electrónica (d_U) Llave pública del generador de medios cifrada ($\hat{}e_{GM}$) Llave publica de resultados (e_R)	Llave privada del equipo que analiza los resultados (d_R) Llave pública de la urna electrónica (e_R)

Tabla 4. Ubicación inicial de las llaves públicas y privadas.

3.3.2 ETAPA DE GENERACIÓN DE MEDIOS.

Una vez que se han generado los archivos de configuración, se debe asegurar que éstos estén protegidos contra el ataque de suplantación. El protocolo para el manejo de la seguridad en esta etapa es:

$s = d_{GM}(a, H(a))$
$k_{ESP} = s_1 + s_2 + s_3 + \ldots + s_n$
$\hat{}e_{GM} = k_{ESP}(e_{GM})$
$c = k_{GM}(a)$
$p = e_U(k_{GM})$

El funcionamiento es el siguiente: Primero se obtiene una firma digital (s) de los archivos que se van a manejar con el objetivo de poder asegurar su autenticidad y su integridad. Posteriormente tomando partes de esas (s_1, s_2, …, s_n) firmas se crea la llave especial (k_{ESP}) y con ella se cifra la llave pública (e_{GM}) que sirve para verificar la integridad y autenticidad de los datos. El siguiente

paso es cifrar los archivos de configuración (a) utilizando la llave simétrica del equipo generador de medios (k_{GM}) que es generada al momento de ser requerida y que se protege cifrándola con la llave pública de la urna electrónica (e_U) lo que garantiza que solo ésta podrá descifrarla. Una vez finalizados estos procedimientos los archivos que se transportan por un canal de comunicaciones inseguro son los datos cifrados (c), la firma digital (s) y la llave simétrica cifrada o password (p).

3.3.3 ETAPA DE VOTACIÓN.

Aquí se trabajó con las etapas de pre votación que es la verificación de la seguridad de los archivos de configuración, votación encargada de la seguridad de los votos y archivos que se generan durante la elección y post votación que maneja la seguridad de los archivos de resultados.

3.3.3.1 Etapa de pre votación.

Lo primero que hay que evitar es que el sistema se encienda en una fecha u hora incorrecta, para esto hay que obtener la hora del sistema y se compararla con una hora de encendido determinada y si la hora del sistema es menor, se reporta que aún no es momento de encender el equipo y éste se apaga. Si la fecha y la hora de encendido son correctas, se procede a la verificación de la integridad de los datos, el protocolo que realiza la verificación de la integridad y autenticidad de los datos es el siguiente:

$k_{GM} = d_U (p)$
$a = k_{GM}^{-1} (c)$
$k_{ESP} = s_1+s_2+\ldots+s_n$
$e_{GM} = k_{ESP}^{-1}(\hat{}e_{GM})$
$e_{GM} (H(a),s)$

Primero se descifra la llave simétrica cifrada (p) utilizando la llave privada de la urna electrónica (d_U) y se obtiene la llave simétrica (k_{GM}) que cifró los datos de configuración. Una vez que se han descifrado éstos datos (c) y se han obtenido los archivos originales (a) se procede a crear, con las firmas de los datos recibidos, la llave especial (k_{ESP}) que descifra la llave pública del generador de medios (e_{GM}) utilizada para verificar la autenticidad de los datos, si las firmas no han sido modificadas la llave creada corresponde a la utilizada para cifrar la llave pública y esta se descifra de manera adecuada lo que permite realizar la verificación de la integridad de los datos.

Una vez que los datos han sido verificados se procede a la configuración del equipo, en esta etapa se tienen los siguientes archivos:

- Datos de configuración.
- Comprobantes de componentes.
- Comprobante de validez de los datos.
- Actas iniciales.

- Actas de apertura.

Para estos archivos la confidencialidad no es tan importante por que la información que contienen es de conocimiento público, lo que se debe proteger es su integridad, los que más riesgo tienen son los datos de configuración ya que son transportados de un lugar a otro mientras que los comprobantes y las actas al generarse dentro del sistema, corren menor riesgo de que alguien tenga acceso a ellos. De todas maneras estos archivos cuentan con cierto nivel de seguridad ya que son cifrados de manera simétrica con una llave de 128 bits de longitud que se genera en el momento en que es requerida, el cifrado se realiza utilizando el algoritmo CAST5.

3.3.3.2 Etapa de votación.

En esta etapa el principal elemento a proteger es el voto almacenado, asegurándose que se cumplan las propiedades de privacidad y correctez que establecen que un voto no debe ser modificado y que no se puede hacer una relación voto - votante, si bien se cuenta con un almacenamiento aleatorio que evita el relacionar un voto con el usuario que lo emitió, una medida para proteger al voto de ser modificado y reforzar su privacidad es el cifrado, cada voto se cifra con su propia llave simétrica de 128 bits utilizando el algoritmo CAST5.

En esta etapa además se cifran también los siguientes elementos:

- Comprobante electrónico.
- Resultados de auditorías intermedias.

Las llaves utilizadas para cifrar los distintos documentos se crean en el momento en que se necesitan y posteriormente se cifran de manera asimétrica utilizando la llave pública del equipo que analiza los resultados, esto como una medida más de seguridad para prevenir que en algún momento pudieran ser extraídos del equipo y modificados.

3.3.3.3 Etapa de post votación.

Una vez que finaliza la jornada electoral, como primera medida de seguridad se cifra la llave simétrica utilizada para el cifrado de las EBI con la llave pública del equipo que analiza los resultados, lo que garantiza que en caso de que quieran ser revisados solo podrán serlo en el lugar donde se encuentre el equipo de análisis de resultados. Posteriormente se genera la seguridad necesaria para el archivo de resultados que será enviado a través de un canal de comunicaciones inseguro. Al igual que en los datos de configuración, proteger la confidencialidad no es tan importante ya que los resultados se conocen e incluso quedan registrados en las actas impresas.

El objetivo del protocolo es depender lo menos posible de la seguridad de las llaves, en especial de las privadas, para la etapa de envío de los resultados se tienen dos opciones.

Opción 1. Capturando los resultados de un acta impresa.

El protocolo para esta opción incluye como última medida de seguridad que los resultados se capturen de un acta que los contenga y en base a esta información poder asegurar que el archivo recibido no ha sido modificado. El protocolo es el siguiente:

d_{Uf}, e_{Uf}
$s = d_{Uf}(r, H(r))$
$c = k_U(r)$
$\hat{}d_{Uf} = k_U(d_{Uf})$
$p = e_R(k_U)$
$k_{ESP} = s(r)$
$\hat{}e_{Uf} = k_{ESP}(e_{Uf})$

Lo primero es crear un juego de llaves asimétricas (d_{Uf} y e_{Uf}) que serán utilizadas para firmar archivo de resultados (r) y obtener su firma digital (s). Posteriormente se cifran el archivo de resultados y la llave privada (d_{Uf}) que se utiliza para generar la firma de los resultados con la llave simétrica (k_U) y ésta se cifra con la llave pública (e_R) del equipo que verifica los resultados. Después se crea una llave especial (k_{ESP}) formada por la firma de los resultados y con ella se cifra la llave pública (e_{Uf}) que verifica la integridad de los datos. Así se obtiene un nuevo conjunto de datos a enviar formado por el archivo de resultados cifrado (c), las llaves pública y privada utilizadas para verificar su integridad y autenticidad cifradas ($\hat{}e_{Uf}$, $\hat{}d_{Uf}$) y la llave simétrica de los resultados cifrada (p).

Opción 2. Envío de la firma a través de un canal de comunicaciones seguro.

En esta opción, se debe enviar la firma digital de los resultados por medio de un canal de comunicaciones considerado seguro. El protocolo es el siguiente:

d_{Uf}, e_{Uf}
$s = d_{Uf}(r, H(r))$
$c = k_U(r)$
$p = e_R(k_U)$
$k_{ESP} = s(r)$
$\hat{}e_{Uf} = k_{ESP}(e_{Uf})$

Primero se deben generar las llaves pública y privada (d_{Uf} y e_{Uf}) para verificar la integridad de los datos. Con la llave privada se obtiene la firma digital (s) del archivo de resultados (r), posteriormente éste se cifra con la llave simétrica (k_U) y ésta se cifra utilizando la llave pública del equipo de análisis de resultados (e_R), luego se crea la llave especial (k_{ESP}) a partir de la firma digital

de los resultados y con ésta se cifra la llave pública (e_{Uf}) que verifica la integridad de los archivos. Así se obtiene un nuevo conjunto de datos a enviar formado por la firma de los resultados (s), el archivo de resultados cifrado (c), la llave pública que verifica la integridad y autenticidad cifrada ($^e_{Uf}$) y la llave simétrica cifrada (p). Aquí el archivo que contiene la firma digital debe ser enviado a través de un canal de comunicaciones seguro junto con algún identificador del equipo que lo generó.

3.3.4 ETAPA DE ANÁLISIS DE RESULTADOS.

En esta etapa el protocolo se encarga de descifrar los archivos recibidos y de verificar su integridad y autenticidad, asegurando detectar si se ha realizado algún cambio en su contenido, al finalizar los datos quedan listos para su análisis. Se cuenta un protocolo por cada una de las opciones de envío generadas en la etapa anterior.

Opción 1. Capturando los resultados del acta final.

El protocolo para esta opción es el siguiente:

$k_U = d_R$ (p)
$r = k_U^{-1}$ (c)
$d_{Uf} = k_U^{-1}$ ($^d_{Uf}$)
$s = d_{Uf}(r_T)$
$k_{ESP} = s$
$e_{Uf} = k_{ESP}(^e_{Uf})$
e_{Uf} (H(r), s)

Al llegar el conjunto de archivos formado por {c, $^e_{Uf}$, $^d_{Uf}$, p} al equipo que recopila y analiza los resultados, lo primero es descifrar la llave asimétrica (k_U) utilizando la llave privada del equipo de análisis de resultados (d_R), después se descifran los resultados (r) y la llave privada que genera la firma digital de los resultados (d_{Uf}). Posteriormente se teclean los resultados contenidos en el acta (r_T) y se crea la firma digital (s) de éstos, con esta firma se forma la llave especial (k_{ESP}) que descifra la llave pública (e_{Uf}) y ésta verifica la integridad de los datos. Una vez que la llave se descifra solo resta verificar la autenticidad del archivo de resultados utilizando la firma que se calculó de los resultados tecleados, éstos al ser iguales a los contenidos en el archivo darán como resultado una verificación válida y se procede a su análisis.

Ventajas.

- No se depende de la seguridad de ninguna llave pública o privada.
- Se pueden enviar los datos a través de un canal de comunicaciones inseguro.
- Es complejo modificar el acta con resultados.

Desventajas.

- Se debe esperar a que llegue el acta al lugar donde se analizan los resultados para poder procesarlos.
- Se debe tener cuidado al momento de teclear los resultados contenidos en el acta para que la firma se genere de manera adecuada, pudiendo resolver este problema con la generación de un código de barras que contenga los resultados.

Opción 2. Envío de la firma a través de un canal de comunicaciones seguro.
El protocolo para esta opción es el siguiente:

$k_U = d_R$ (p)
$r = k_U^{-1}$ (c)
$k_{ESP} = s$
$e_{Uf} = k_{ESP}(\hat{}e_{Uf})$
e_{Uf} (H(r), s)

Al llegar el conjunto de archivos formado por {c, $\hat{}e_{Uf}$, p} al equipo que recopila y analiza los resultados y una vez que se tenga la firma digital {s}, el procedimiento es el siguiente:

Lo primero es descifrar la llave simétrica (k_U) utilizando la llave privada del equipo de análisis de resultados (d_R), y con ella se descifran los resultados (r), con la firma digital (s) recibida se crea la llave especial (k_{ESP}) que descifra la llave pública (e_{Uf}) que verifica integridad del archivo de resultados.

Ventajas.

- Se envían menos datos.
- No se tiene que esperar a que el acta llegue y que los datos se escriban de manera correcta.

Desventajas.

- Se debe tener un canal de comunicaciones seguro.
- La seguridad depende de que nadie obtenga la firma y la substituya por una propia.

Las Figuras 7 y 8 muestran respectivamente el funcionamiento del protocolo en las tres etapas y considerando las dos diferentes opciones para el envío de resultados al equipo analizador de resultados.

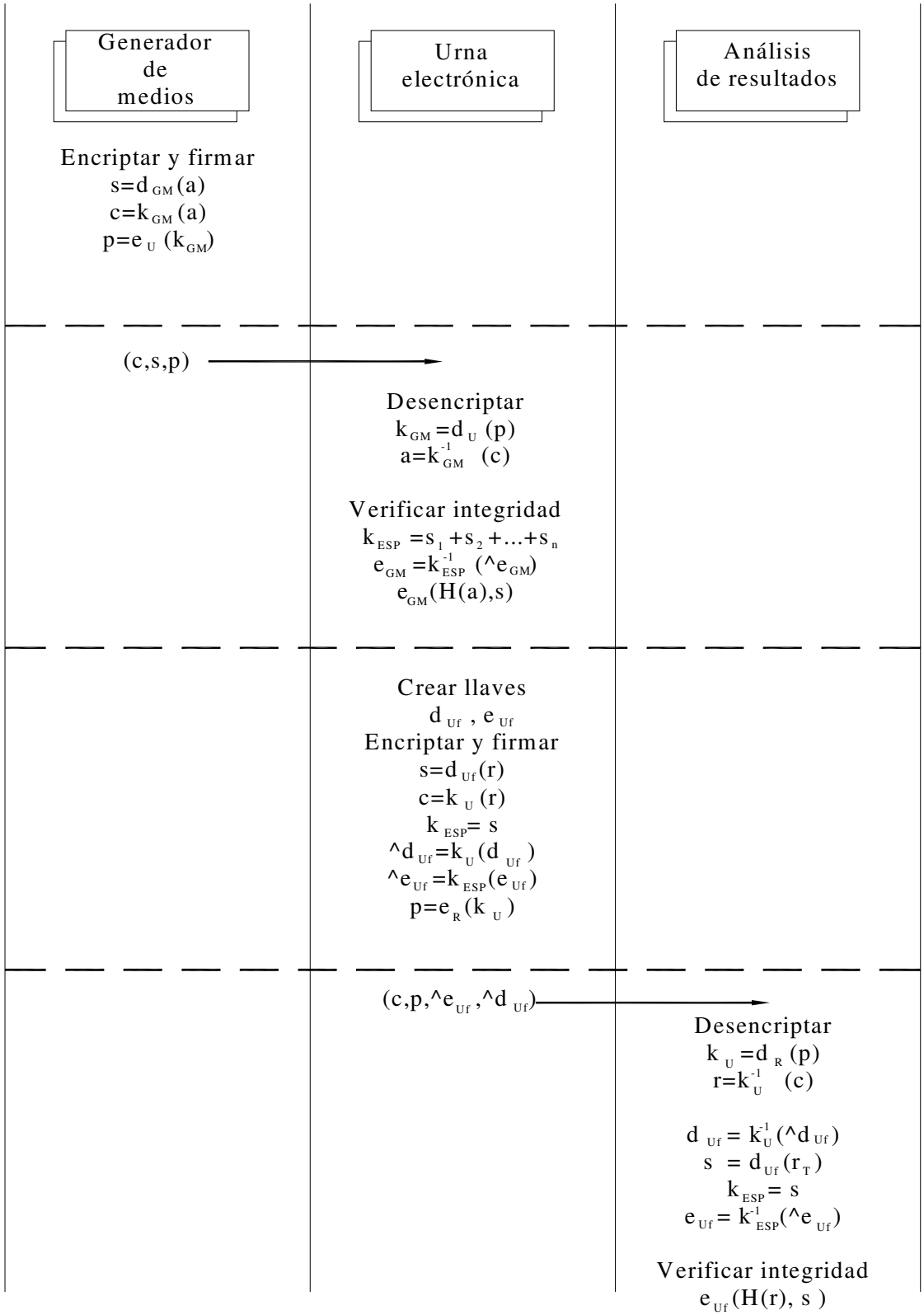


Figura 7. Funcionamiento del protocolo criptográfico para la Opción 1.

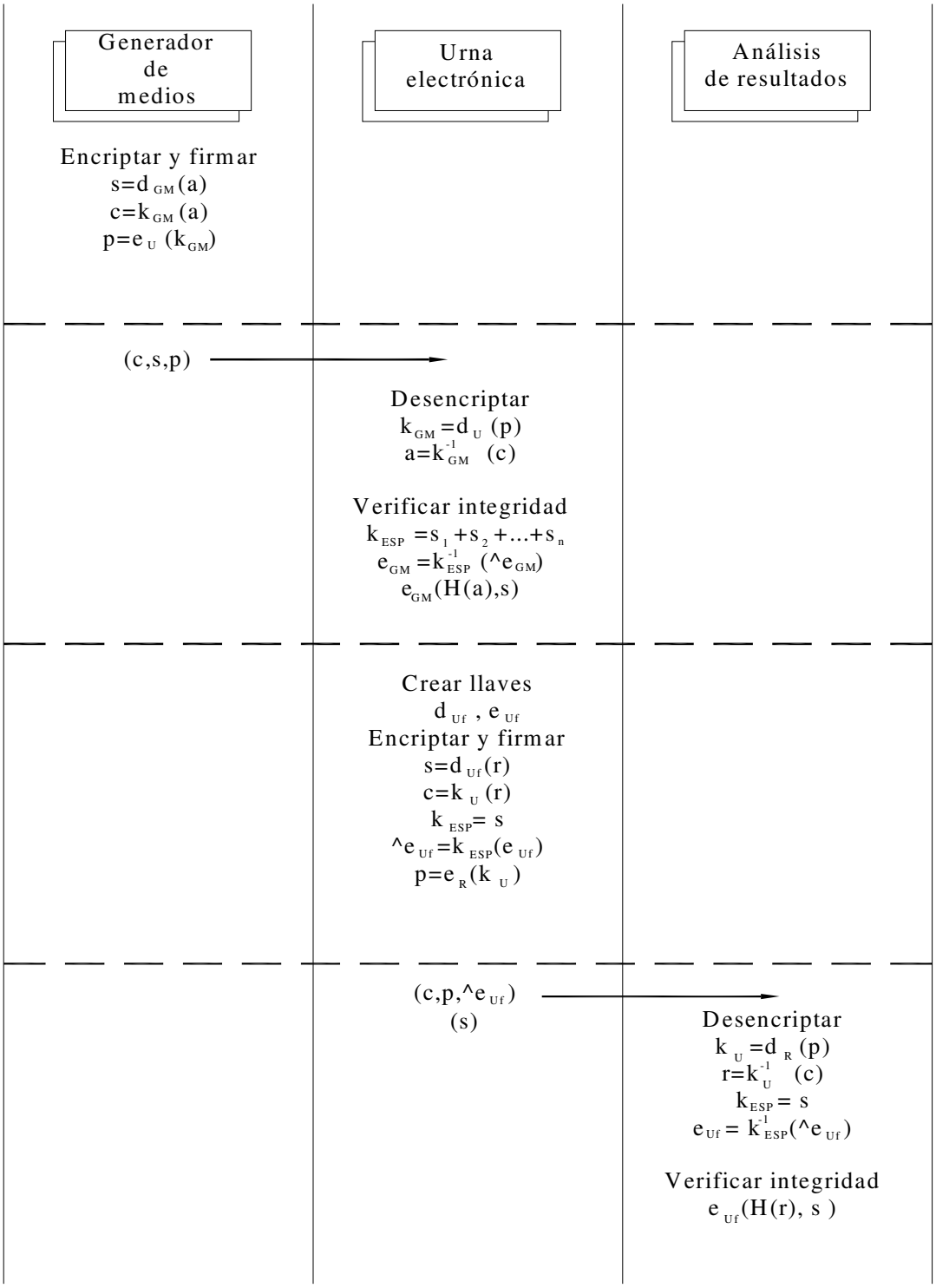


Figura 8. Funcionamiento del protocolo criptográfico para la Opción 2.

3.4 DESARROLLO DE LA ARQUITECTURA.

Con base en lo revisado en los antecedentes acerca de la creación de arquitecturas de seguridad, el desarrollo de la arquitectura de seguridad se realizó de la siguiente manera.

Se tomaron los distintos niveles que menciona Probst [Probst, 2002] y se adaptaron a las necesidades del sistema, éste al ser un sistema de voto presencial no requería de la parte de comunicación segura por lo que se eliminó, y se cambio el orden de las capas de Autenticación y Control de acceso, así que los elementos que se consideraron en la arquitectura de seguridad fueron:

- Autenticación.
- Control de Acceso.
    - Sujeto.
    - Objeto seguro.
    - Autorización.
    - Restricciones.
- Criptografía.

[Probst, 2002] menciona a la auditoría como un elemento más de esta arquitectura, pero se trabajó de manera diferente al ser el objetivo del proyecto tener una arquitectura de seguridad y de auditoría, por lo que la auditoría se encuentra presente en cada una de las capas. Para la construcción de la arquitectura se trabajó con las distintas etapas que forman el proceso de votación en el sistema, diseñando una arquitectura para cada una de ellas.

3.4.1 Arquitectura de la etapa de pre votación.

Aquí no se incluyó la parte de autenticación, sólo el control de acceso, la criptografía y la auditoría.

El control de acceso está formado por los siguientes elementos:
Sujeto: El usuario que enciende el equipo.
Objeto seguro: La interfaz de configuración.
Autorización: El permitir el acceso a la etapa de configuración.
Restricciones: La única restricción es la fecha y hora de encendido que debe ser posterior a la que el sistema tiene registrada para comenzar su funcionamiento.

La criptografía está encargada del cifrado de los distintos archivos que se generen. La Figura 9 muestra las capas de esta arquitectura así como los elementos tanto de seguridad como de auditoría que se encuentran en cada una de ellas.

Seguridad	Auditoria
Control de acceso. (Fecha y hora de encendido.)	(Bitácora.)
Criptografía. (Validación de la autenticidad de los datos de configuración. Cifrado de los elementos generados en la parte de auditoría.)	Bitácora. (Comprobantes generados : Funcionamiento de los componentes. Comprobante de validez de los datos. Acta de apertura. Actas iniciales.)

Figura 9. Arquitectura de seguridad y auditoría de la etapa de pre votación.

Una vez que se tienen los elementos que conforman cada capa de la arquitectura es importante conocer como interactúan entre ellos durante el proceso que se lleva cabo en esa etapa, esto se muestra en la Figura 10.

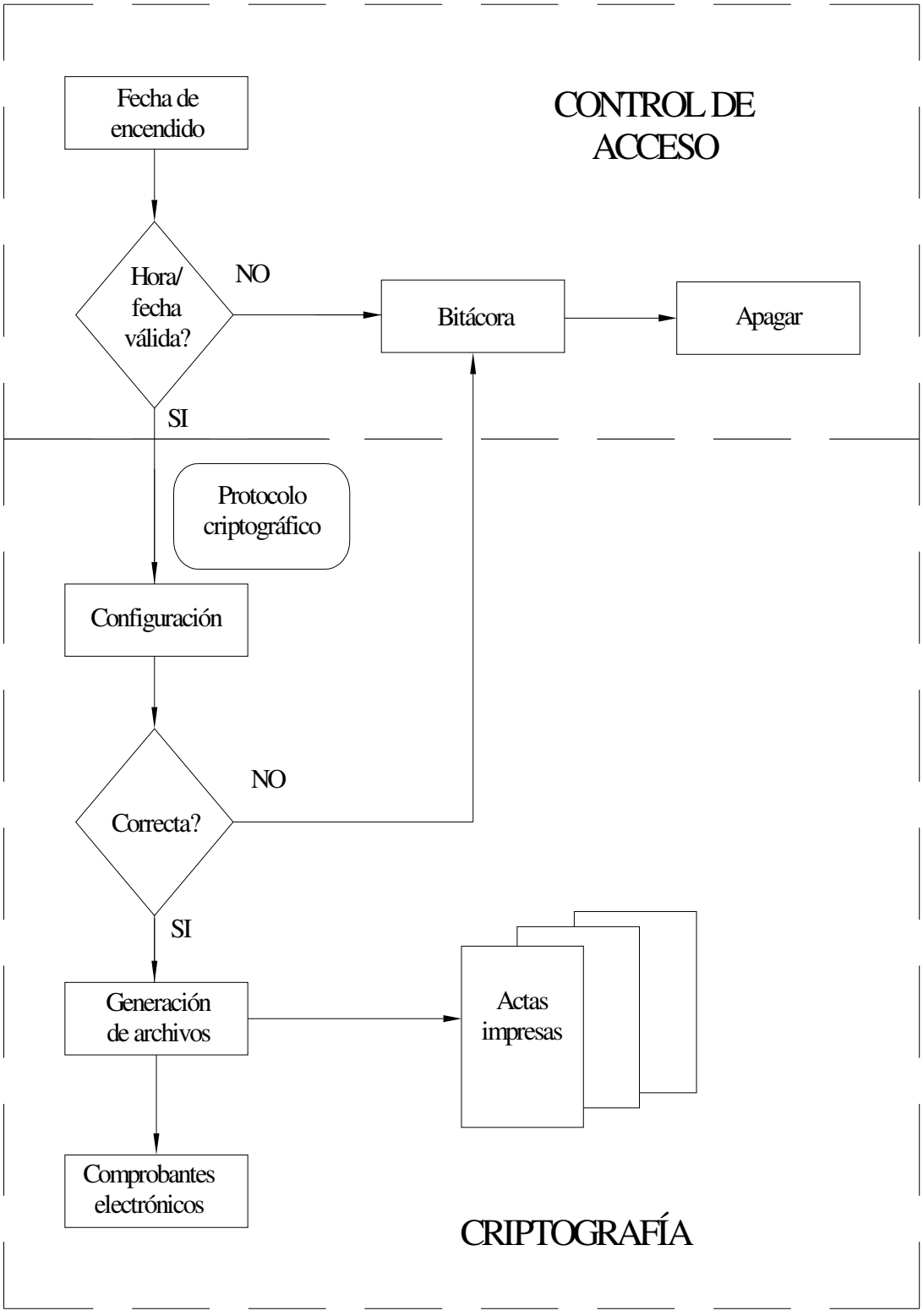


Figura 10. Interacción de los elementos de la arquitectura en la etapa de pre votación.

3.4.2 Arquitectura de la etapa de votación.

En esta etapa el método de autenticación consiste en comparar código de barras esperado contra el introducido, ya que ésta es la forma en que se activa el sistema en donde se realizan las pruebas, si coinciden entonces comienza la tarea del control de acceso.

El control de acceso se basa en decidir qué tipo de usuario se encuentra presente, si es un usuario normal o un administrador, al ser la etapa de votación el usuario será un votante normal, los componentes del control de acceso son:

Sujeto: Usuario (Votante).
Objeto seguro: Interfaz de votación que permite el acceso al archivo de votos.
Autorización: Acceso al archivo de votos para poder añadir información.
Restricciones: La restricción es que el votante solo puede participar una vez, ya que terminó sus elecciones el sistema no deberá permitir que éste vote nuevamente. Otras restricciones que podrían existir son relativas al tiempo, por ejemplo que el sistema no permita que el votante vote si se ha tardado más de un determinado número de minutos en realizar su elección o que el sistema no permita más votos después de cierta hora. La Figura 11 muestra las capas de esta arquitectura y sus elementos.

Seguridad	Auditoría
Autenticación. (Comparación entre el código esperado y el leído.)	(Bitácora.)
Control de acceso. (Votante. Administrador.)	(Bitácora. Deshabilitación del equipo una vez que el usuario ha terminado de votar.)
(Criptografía. Cifrado de los votos individuales. Descifrado de los votos individuales para conteos parciales. Votos almacenados de manera aleatoria. Cifrado de los elementos producidos en la auditoría.)	(Almacenamiento en diversos medios. Comprobantes generados : Comprobante impreso. Comprobante electrónico. Archivos de resultados de las auditorías intermedias.)

Figura 11. Arquitectura de seguridad y auditoría de la etapa de votación.

La manera en la que interactúan las distintas capas y sus elementos durante el proceso de votación se muestra en la Figura 12.

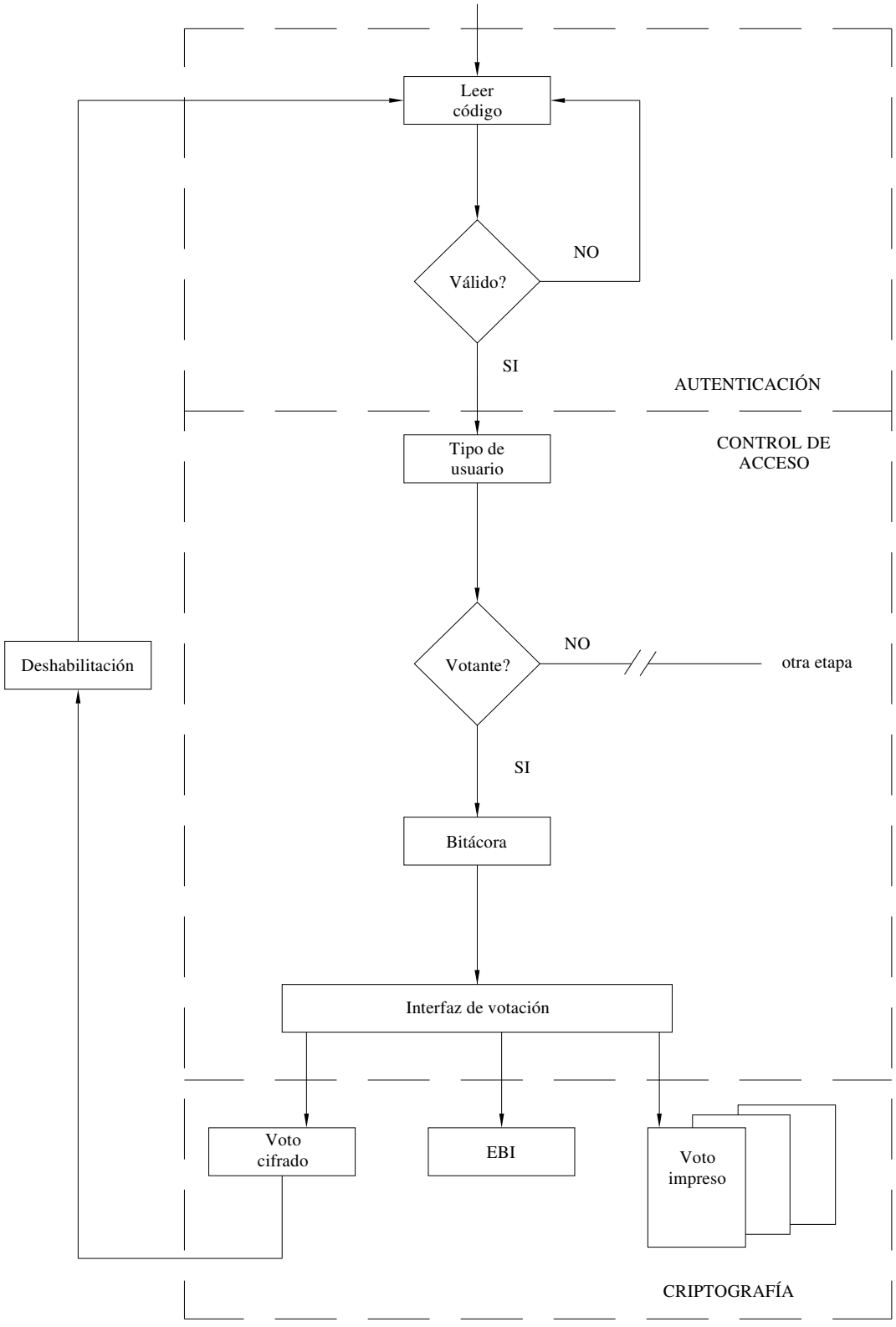


Figura 12. Interacción de los elementos de la arquitectura en la etapa de votación.

3.4.3 Arquitectura de la etapa de post votación.

El método de autenticación es el mismo que en la etapa anterior sólo que esta vez el usuario sería un administrador y no un votante.

El control de acceso se basa en decidir qué tipo de usuario se encuentra presente, si es un usuario normal o un administrador, al ser la etapa de post votación el usuario tendría que ser un administrador, los componentes del control de acceso son:

Sujeto: Usuario (Administrador).
Objeto seguro: Interfaz de administración que permite el acceso al archivo con los votos.
Autorización: Acceso al archivo de votos para poder leer la información y realizar el conteo final.
Restricciones: No se tiene alguna restricción en especial, pudiendo existir una restricción relacionada con el tiempo, por ejemplo que el sistema no permita entrar al módulo de administración sino hasta después de determinada hora.

La criptografía está encargada de cifrar los archivos que se generen incluyendo el que contiene los resultados finales en un formato específico. La Figura 13 muestra la arquitectura para esta etapa y los elementos que se encuentran en cada capa.

Seguridad	Auditoría
Autenticación. (Comparación entre el código esperado y el leído.)	(Bitácora.)
Control de acceso. (Votante. Administrador.)	(Bitácora. Apagado del equipo.)
Criptografía (Descifrado de los votos individuales para el conteo. Cifrado de los elementos producidos en la auditoria. Protocolo criptográfico.)	(Archivo de resultados finales. Comprobantes generados : Actas finales. Acta de cierre de votación. Archivo de resultados.)

Figura 13. Arquitectura de seguridad y auditoría de la etapa de post votación

La manera en que se relacionan los distintos elementos de cada capa en esta arquitectura se muestra en la Figura 14.

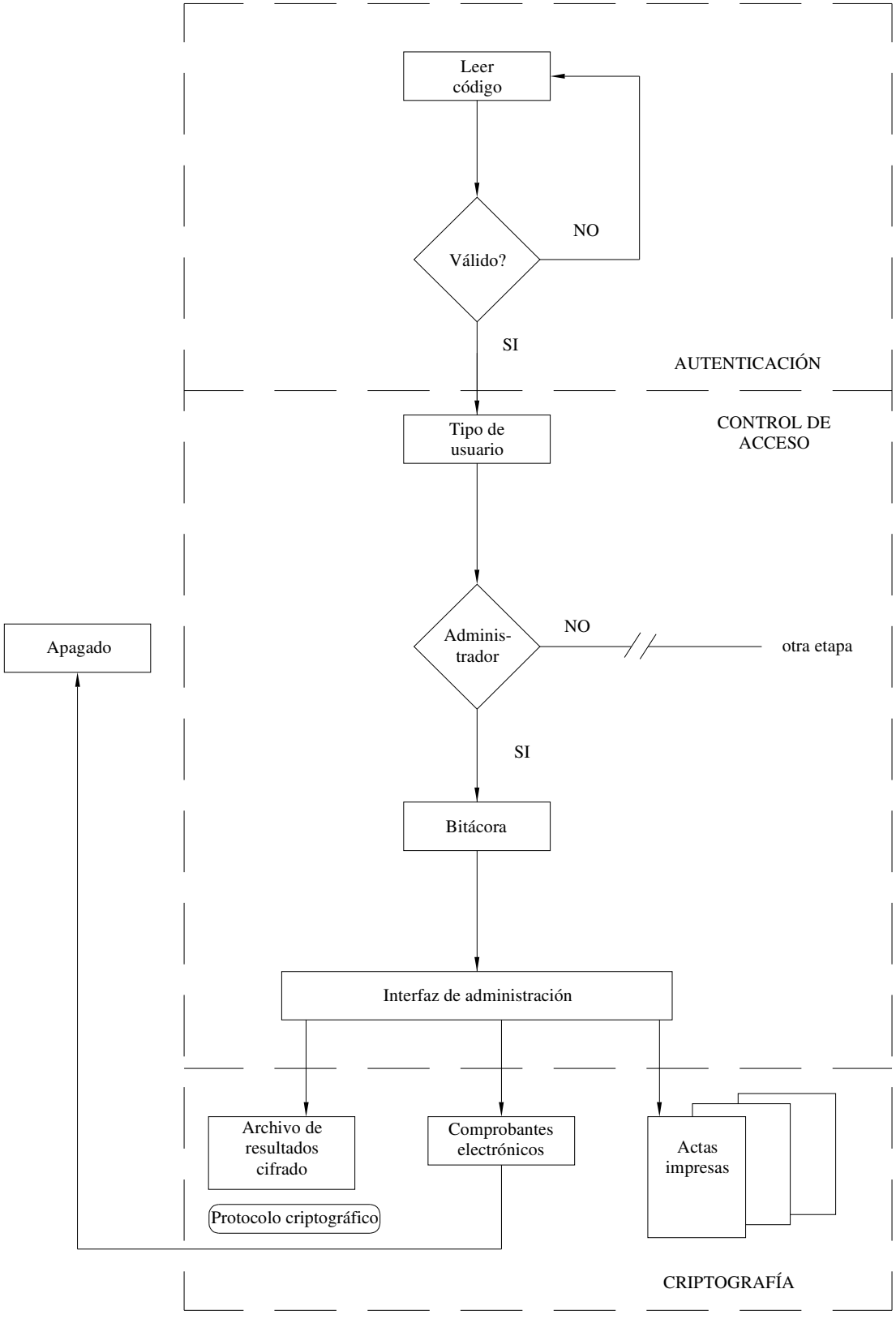


Figura 14. Interacción de los elementos de la arquitectura en la etapa de post votación.

CAPÍTULO 4

PRUEBAS Y RESULTADOS.

En este capítulo se presentan las pruebas realizadas a la auditoría, las cuales están relacionadas con el uso del sistema y la confianza que genera hacia el público en general. También las pruebas hacia la seguridad que consistieron en revisar los protocolos criptográficos analizando los distintos casos que se tienen según el acceso a diferentes llaves que se consideran seguras, también se revisaron algunos elementos que debe cumplir un sistema de voto electrónico y cómo se manejaba en esos casos la seguridad. Para el código fuente se presentan los resultados y el análisis de la revisión que se hizo con los programas *Flawfinder* y RATS.

4.1 PRUEBAS A LA AUDITORÍA.

Para probar el funcionamiento general del sistema y la confianza que genera hacia los usuarios se realizó una prueba de campo, de acuerdo al manejo de [Jones, 2004B] sobre las pruebas paralelas y los recursos disponibles de tiempo, equipo y espacio se eligieron las siguientes opciones.

El día de la elección la prueba se realizó:

- En otra locación

El equipo se seleccionó:

- Por adelantado

Los votos fueron introducidos:

- Por el público en general
- Por voluntarios entrenados

Las pruebas fueron:

- No dirigidas

El sitio de las pruebas estuvo abierto:

- Durante un horario reducido

Los votos registrados fueron tabulados:

- A mano en el lugar de la prueba

Para las salidas que el sistema generó durante la prueba:

- Se contaron a mano los votos y se comprobaron las actas finales con la tabulación
- Los registros fueron examinados en el mismo equipo

Para que los usuarios, personal académico y alumnos de la UAM-A tuvieran un conocimiento claro de las opciones que se podían elegir, se realizó una encuesta donde se preguntaba sobre la calificación que los usuarios le darían a distintos servicios que se tienen en la universidad como son:

- Biblioteca.
- Cafetería.
- Servicios administrativos.
- Centro de cómputo.

Y las posibles opciones para calificar fueron:

- MB (Muy bien).
- B (Bien).
- S (Suficiente).
- NA (No aprobado).
- Voto el blanco (Sin opinión).

Adicionalmente se preparó un cuestionario para que los usuarios opinaran sobre el funcionamiento general y la confianza en el sistema de votación electrónica utilizado. El cuestionario era el siguiente:

---

1. El uso del equipo te pareció:

( ) Muy simple
( ) Simple
( ) Complejo
( ) Muy complejo

2. Confías en que el sistema registró la opción que elegiste:

( ) Si
( ) No
( ) Sí hasta conocer los resultados
( ) No estoy seguro

3. El ver que los votos totales en las elecciones se incrementan después de tu participación hace que tu confianza en el sistema:

( ) Aumente
( ) Siga igual
( ) Disminuya
( ) No lo noté

4. El comprobante impreso con la opción que escogiste hace que tu confianza en el sistema:

( ) Aumente
( ) Siga igual
( ) Disminuya
( ) No lo noté

5. Si la seguridad de un sistema de votación es que tu voto no pueda ser modificado, el voto electrónico te parece:

( ) Muy seguro
( ) Seguro
( ) Inseguro
( ) Muy inseguro

6. Si conocieras la forma en que esta elaborado el sistema y las medidas de seguridad con las que cuenta, tu confianza en él:

( ) Aumentaría
( ) Seguiría igual
( ) Disminuiría
( ) No estoy seguro

7. Comparado con el voto tradicional, el voto electrónico te parece:

( ) Mucho más sencillo

( ) Más sencillo

( ) Más complejo

( ) Mucho más complejo

8. Comparado con el voto tradicional, el voto electrónico te parece:

( ) Mucho más confiable

( ) Más confiable

( ) Igual de confiable

( ) Menos confiable

9. Comparado con el voto tradicional, realizar fraudes en el voto electrónico es:

( ) Mucho más sencillo

( ) Más sencillo

( ) Más complejo

( ) Mucho más complejo

COMENTARIOS.

En la prueba participaron 65 usuarios y los resultados que se obtuvieron se presentan en las siguientes figuras.

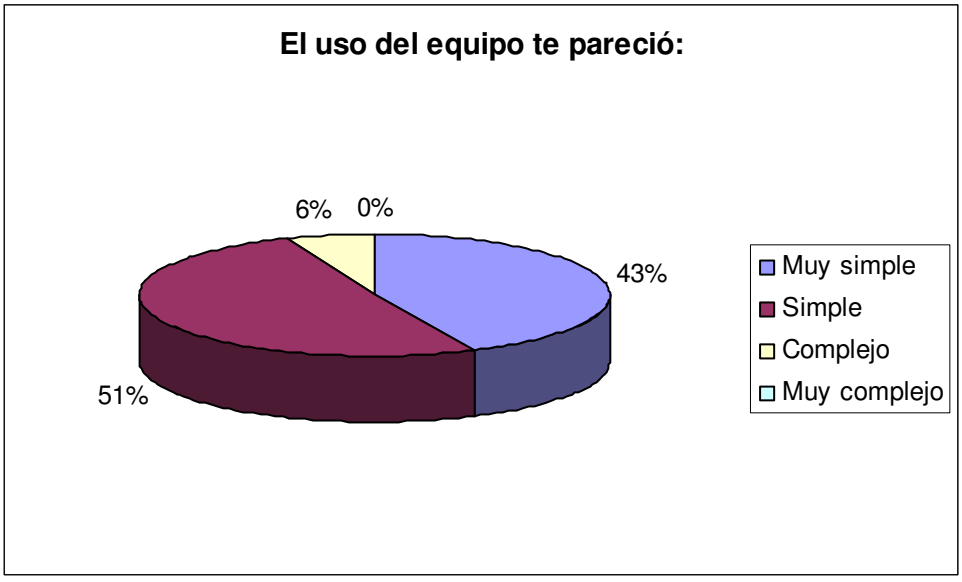


Figura 15. Uso del equipo.

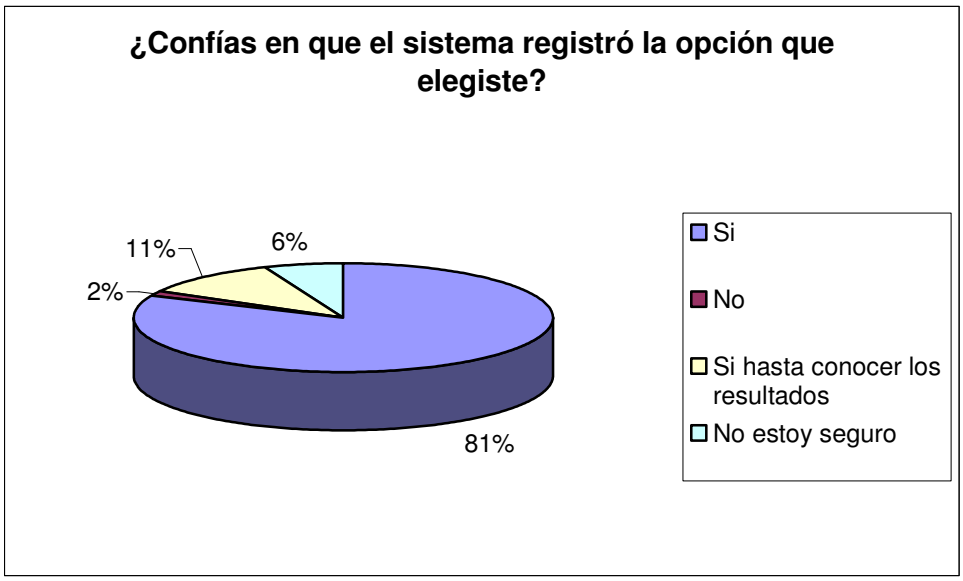


Figura 16. Confianza en el registro del voto.

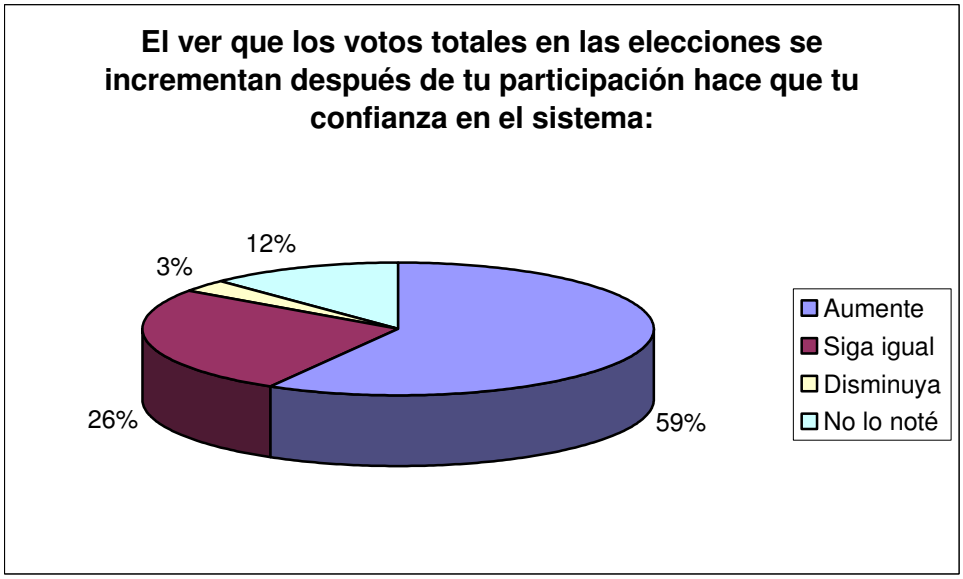


Figura 17. Efecto del incremento de los votos registrados en el nivel de confianza.

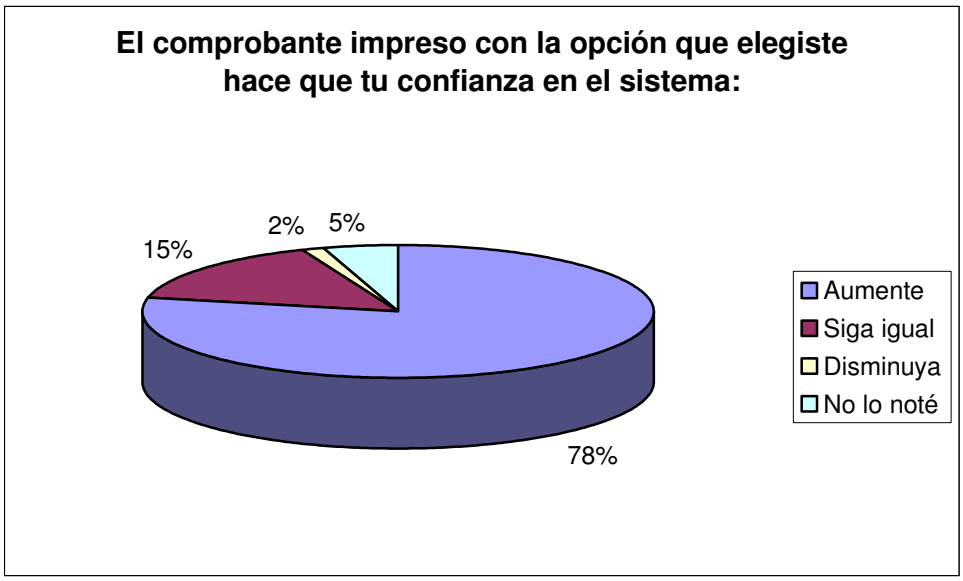


Figura 18. Efecto del comprobante impreso en el nivel de confianza.

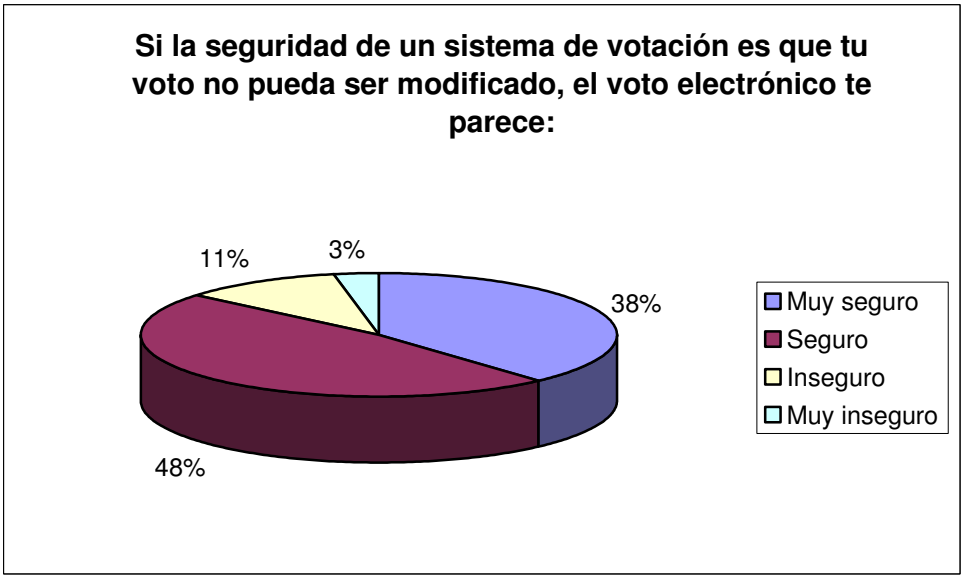


Figura 19. Nivel de seguridad del sistema de voto electrónico probado.

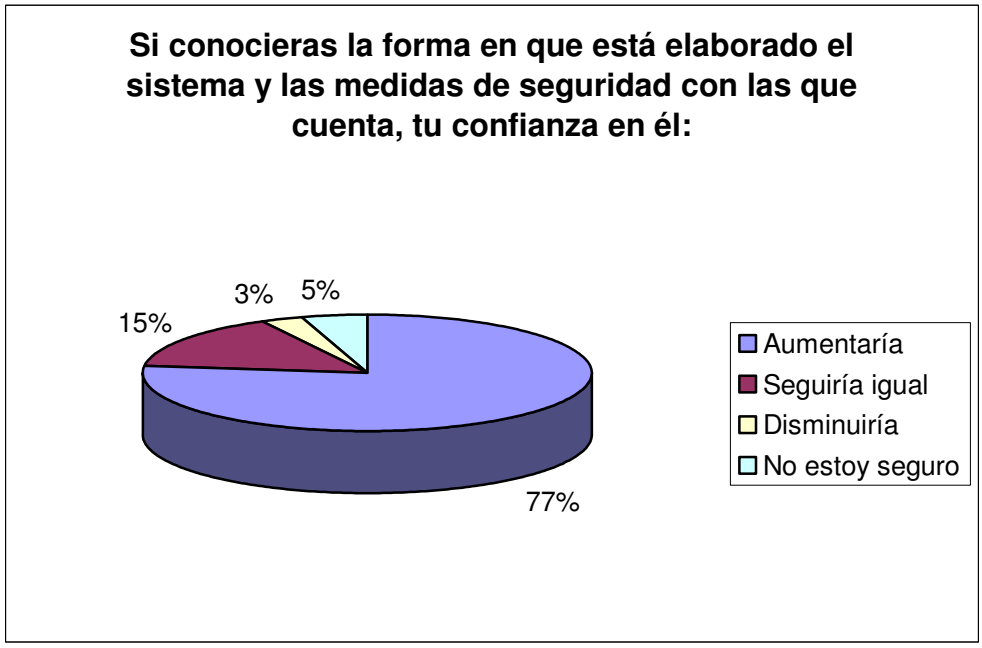


Figura 20. Efecto del conocimiento de la manera en que se elaboró el sistema en el nivel de confianza.

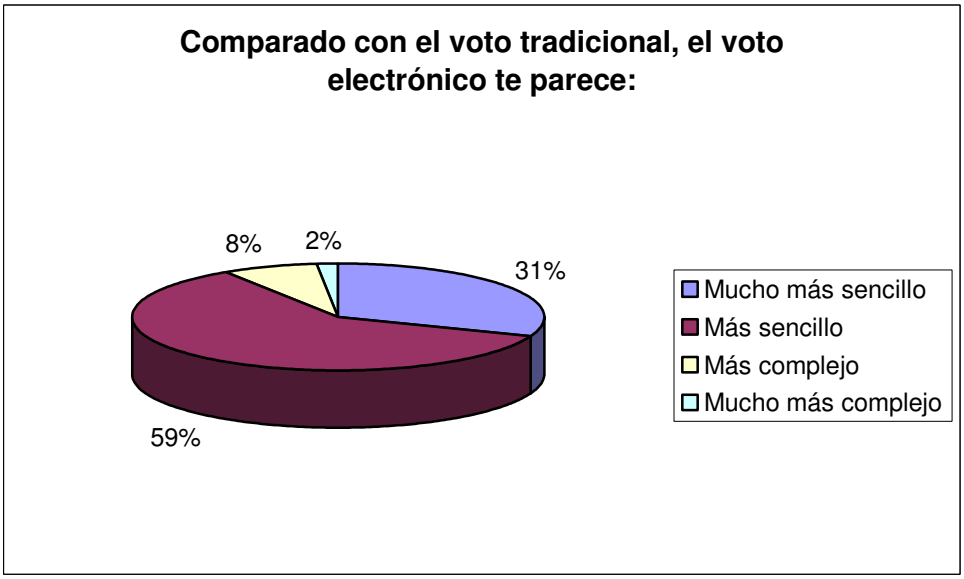


Figura 21. Comparación del uso entre el voto tradicional y el electrónico.

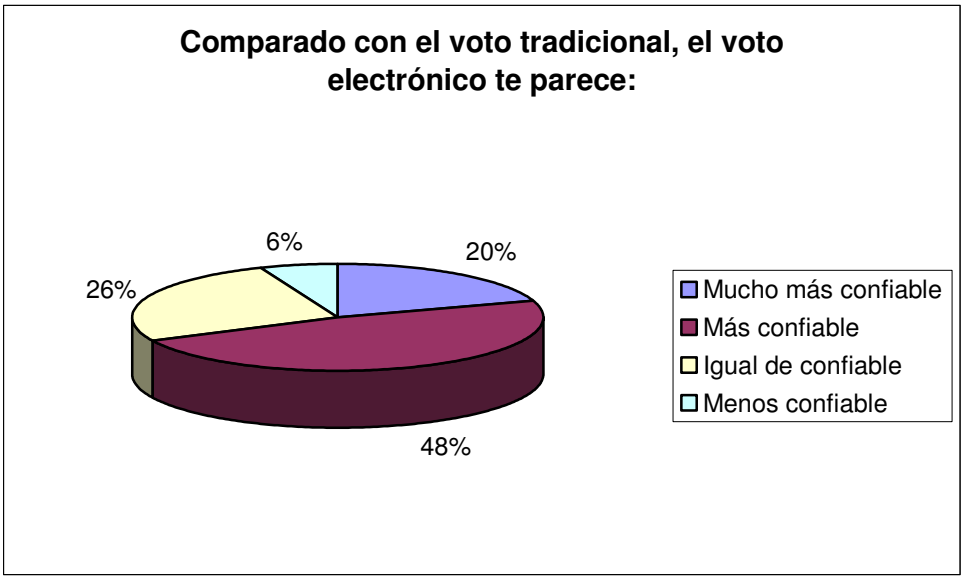


Figura 22. Comparación de la confiabilidad entre el voto tradicional y el electrónico.

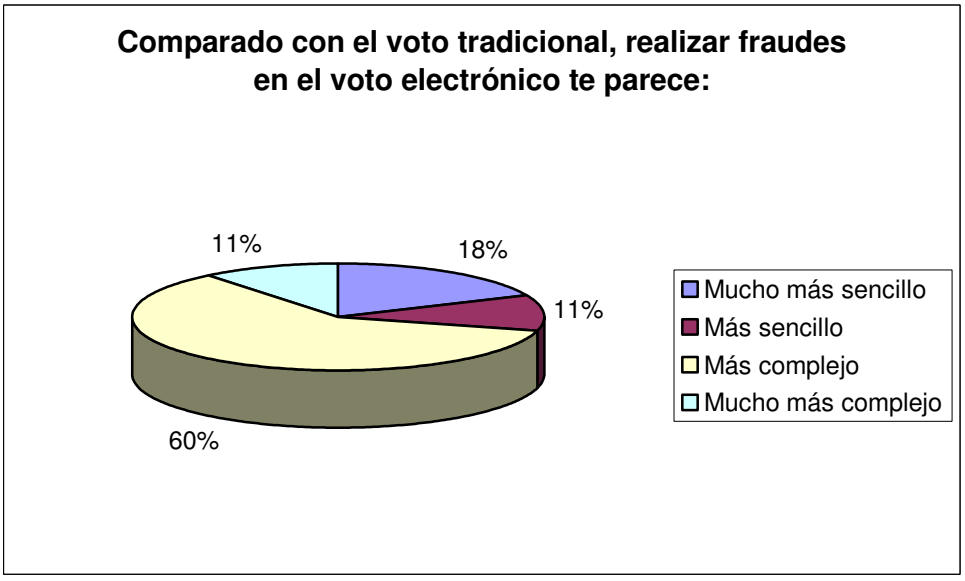


Figura 23. Comparación de la seguridad entre el voto tradicional y el electrónico.

Los resultados muestran que el uso del equipo se considera entre muy sencillo y sencillo. Un gran porcentaje confió en que el sistema registró la opción que eligieron y otros sólo hasta conocer los resultados mientras que un porcentaje muy pequeño no confiaba en él. El ver que los votos se incrementaban después de finalizada la participación de cada usuario también demostró generar un buen aumento en la confianza, aunque para algunos permaneció igual y otros no lo notaron, el comprobante impreso genera un aumento en la confianza mayor que el incremento de los votos y es más sencillo de notar al darse cuenta de que la impresora emite algún sonido. El voto electrónico es considerado en un gran porcentaje entre seguro y muy seguro, el conocer como está realizado el sistema hace que la confianza en él aumente. Finalmente en las comparaciones del voto electrónico con el voto tradicional el primero se considera más sencillo, más confiable y donde parece tener cierta desventaja es en la facilidad de realizar fraudes, ya que si bien el porcentaje de personas que consideran que es más complicado realizar fraudes es alto, el porcentaje de usuarios que opinan que es mucho más sencillo alterar los resultados es considerable.

Estos resultados muestran que los elementos auditables de la arquitectura que el usuario puede ver como el comprobante impreso y el reporte de la contribución de su voto hacen que su confianza se incremente, por lo que su adición a la misma es necesaria.

4.2 PRUEBAS A LA SEGURIDAD.

La primera prueba fue al protocolo criptográfico que se encuentra entre el equipo generador de medios y la urna electrónica. Los algoritmos de cifrado son considerados muy seguros ya que no se pueden alterar los datos tratando de romper la llave con la que fueron cifrados esto de acuerdo a la longitud de llaves mostrada en la Tabla 5 [García et al, 2004].

Nivel de seguridad	Longitud de llave simétrica	Longitud de llaves públicas.
Aceptable (de 5 a 10 años)	80 bits	1024 bits
Buena (posiblemente para siempre)	128 bits	2048 bits
Extrema	192 bits	4096 bits
Muy extrema	256 bits	8192 bits

Tabla 5. Seguridad y longitud de llaves.

Las pruebas consistieron en crear un nuevo conjunto de datos de configuración y de resultados, generar nuevos juegos de llaves y probar los distintos casos de posesión de las llaves originales por parte de los atacantes. La nomenclatura utilizada en las pruebas es la misma que en el desarrollo pudiéndose consultar en la página 44.

4.2.1 GENERADOR DE MEDIOS A URNA ELECTRÓNICA.

La Tabla 6 muestra los elementos originales, los modificados y los resultados que se obtuvieron en esta etapa.

Caso	Datos originales	Datos modificados	Resultado de la verificación
1	---	d_{GM}, e_U, s, a	ERROR
2	e_U	d_{GM}, s, a	ERROR
3	e_U, s	d_{GM}, a	ERROR
4	e_U, d_{GM}	s, a	ERROR
5	e_U, d_{GM}, s	a	ERROR
6	e_U, d_{GM}, s, a	---	ÉXITO

Tabla 6. Resultados de la prueba para la etapa de generador de medios a la urna.

Caso 1. Suplantación de todos archivos y las llaves.

Datos originales: ---

Datos modificados: d_{GM}, e_U, s, a.

Generación:

*s = *d_{GM} (*a) Se firman los datos modificados.

$*c = k_{GM}(*a)$	Se cifran los datos.
$*p = *e_U(*k_{GM})$	Se cifra la llave simétrica.
Transporte (*s, *c, *p)	
Verificación:	
$*k_{GM} \neq d_U(*p)$	**ERROR**: No coincide la llave pública creada por el atacante con la llave privada original.

Caso 2. Se conoce la llave pública original, el resto de los datos son modificados

Datos originales: e_U

Datos modificados: d_{GM}, s, a.

Generación:	
$*s = *d_{GM}(*a)$	Se firman los datos modificados.
$*c = k_{GM}(*a)$	Se cifran los datos
$*p = e_U(*k_{GM})$	Se cifra la llave simétrica.
Transporte (*s, *c, *p)	
Verificación.	
$*k_{GM} = d_U(*p)$	Coincide la llave pública con la llave privada.
$*a = *k_{GM}^{-1}(*c)$	Los datos se descifran correctamente.
$e_{GM} \neq *k_{ESP}^{-1}(\hat{}e_{GM})$	**ERROR:** La llave pública no se descifra de manera correcta por que la firma modificada crea una llave especial diferente a la esperada

Caso 3. Se conocen la llave pública y la firma original, el resto de los datos se modifican.

Datos originales: e_U, s.

Datos modificados: d_{GM}, a.

Generación:	
$*s = *d_{GM}(*a)$	Se firman los datos modificados.
$*c = k_{GM}(*a)$	Se cifran los datos.
$*p = e_U(*k_{GM})$	Se cifra la llave simétrica.
Transporte (s, *c, *p)	
Verificación:	
$*k_{GM} = d_U(*p)$	Coincide la llave pública con la llave privada.
$*a = *k_{GM}^{-1}(*c)$	Los datos se descifran correctamente.
$e_{GM} = k_{ESP}^{-1}(\hat{}e_{GM})$	La llave pública se descifra de manera correcta.
$e_{GM}(*a, s)$	**ERROR:** La verificación no es válida por que los datos modificados no corresponden con la firma original.

Caso 4. Se conocen las llaves pública y privada.

Datos originales: e_U, d_{GM}.

Datos modificados: s, a.

Generación:

$*s = d_{GM}(*a)$	Se firman los datos modificados.
$*c = *k_{GM}(*a)$	Se cifran los datos.
$*p = e_U(*k_{GM})$	Se cifra la llave simétrica.

Transporte (*s, *c, *p)

Verificación:

$k_{GM} = d_U(*p)$	Coincide la llave pública con la llave privada.
$*a = k_{GM}^{-1}(*c)$	Los datos se descifran correctamente.
$e_{GM} \neq k_{ESP}^{-1}(\hat{}e_{GM})$	**ERROR:** La llave pública no se descifra de manera correcta por que la firma modificada no genera la llave especial correcta.

Caso 5. Solo se modifican los archivos de configuración, el resto de los datos y firmas son los originales.

Datos originales: e_U, d_{GM}, s.

Datos modificados: a.

Generación:

$*s = d_{GM}(*a)$	Se firman los datos modificados.
$*c = *k_{GM}(*a)$	Se cifran los datos.
$*p = e_U(*k_{GM})$	Se cifra la llave simétrica.

Transporte (s, *c, *p)

Verificación:

$k_{GM} = d_U(*p)$	Coincide la llave pública con la llave privada.
$*a = k_{GM}^{-1}(*c)$	Los datos se descifran correctamente.
$e_{GM} = k_{ESP}^{-1}(\hat{}e_{GM})$	La llave pública se descifra de manera correcta por que la firma original genera la llave correcta.
$e_{GM}(*a, s)$	**ERROR:** La verificación no es válida por que los datos modificados no coinciden con la firma original.

Caso 6. Caso correcto donde no se modifica ningún dato y se usan las llaves originales.

Datos originales: e_U, d_{GM}, s, a.

Datos modificados:---

Generación:

$s = d_{GM}(a)$	Se firman los datos originales.
$c = k_{GM}(a)$	Se cifran los datos.
$p = e_U(k_{GM})$	Se la llave simétrica.

Transporte (s, c, p)

Verificación:

$k_{GM} = d_U$ (p)	Coincide la llave pública con la llave privada.
$a = k_{GM}^{-1}$ (c)	Los datos se descifran correctamente.
$e_{GM} = k_{ESP}^{-1}(\hat{}e_{GM})$	La llave pública se descifra de manera correcta por que la firma original genera la llave correcta.
$e_{GM}(a, s)$	**ÉXITO:** La verificación es válida por que los datos coinciden con la firma original.

4.2.2 URNA ELECTRÓNICA.

La urna electrónica incluye una parte de las pruebas anteriores, la verificación de la autenticidad y de la integridad de los datos se realiza dentro de la misma, en la etapa de votación sólo se consideraron las propiedades de los algoritmos de cifrado ya que es la parte que menos riesgos corre al no tener algún modo de acceder a la información que contiene, la seguridad incluía el cifrado de los diferentes documentos y de los votos además del almacenamiento aleatorio que no permitía una relación voto-votante, una vez que termina el proceso, los elementos que se generan de acuerdo al protocolo se prueban en otro equipo que es lo que se tendría en un caso real.

4.2.3 URNA ELECTRÓNICA A EQUIPO DE ANALIZADOR DE RESULTADOS

Aquí se revisa el funcionamiento del protocolo para las dos opciones que se tienen para enviar los resultados al equipo que los analizará.

Opción 1. Capturando los resultados del acta final.

Los resultados de las pruebas para el envío de datos de la urna electrónica al equipo analizador de resultados utilizando la opción de capturar los resultados contenidos en un acta final se muestran en la Tabla 7.

Caso	Datos originales	Datos modificados	Resultado de la verificación
1	---	e_{Uf}, e_R, d_{Uf}, s, r	ERROR
2	e_{Uf}, e_R	s, r, d_{Uf}	ERROR
3	e_{Uf}, e_R, d_{Uf}	s, r	ERROR
4	e_{Uf}, e_R, d_{Uf}, s	r	ERROR
5	e_{Uf}, e_R, d_{Uf}, s, r	---	ÉXITO

Tabla 7. Resultados para la opción 1 de la urna al equipo analizador de resultados.

Caso 1. Suplantación de todas las llaves y los datos.

Datos originales: ---

Datos modificados: s, r, d_{Uf}, e_{Uf}, e_R.

Generación:

$*s = *d_{Uf}(*r)$	Se firman los resultados modificados.
$*c = k_U(*r)$	Se cifran los resultados.
$\hat{}d_{Uf} = k_U(*d_{Uf})$	Se cifra la llave privada.
$k_{ESP} = *s$	Se crea la llave especial.
$\hat{}e_{Uf} = k_{ESP}(*e_{Uf})$	Se cifra la llave pública.
$p = *e_R(k_U)$	Se cifra la llave simétrica.

Transporte ($*c$, $\hat{}d_{Uf}$, $\hat{}e_{Uf}$, $*p$)

Verificación:

$k_U \neq d_R$ (p).	**ERROR:** La llave pública no corresponde con la privada.

Caso 2.Uso de las llaves públicas originales, el resto de los datos se modifica.

Datos originales: e_{Uf}, e_R

Datos modificados: s, r, d_{Uf}.

Generación:

$*s = *d_{Uf}(*r)$	Se firman los resultados modificados.
$*c = k_U(*r)$	Se cifran los resultados.
$\hat{}d_{Uf} = k_U(*d_{Uf})$	Se cifra la llave privada.
$k_{ESP} = *s$	Se crea la llave especial.
$\hat{}e_{Uf} = k_{ESP}(e_{Uf})$	Se cifra la llave pública.
$p = e_R(k_U)$	Se cifra la llave simétrica.

Transporte ($*c$, $\hat{}d_{Uf}$, $\hat{}e_{Uf}$, $*p$)

Verificación:

$k_U = d_R$ (p)	La llave pública corresponde con la privada.
$*r = k_U^{-1}(c)$	Los resultados se descifran de manera correcta.
$d_{Uf} = k_U^{-1}(\hat{}d_{Uf})$	La llave privada se descifra de manera correcta.
$s = d_{Uf}(r_T)$	Se leen los resultados del acta y se firman.
$k_{ESP} = s$	Se crea la llave especial.
$e_{Uf} \neq k_{ESP}(\hat{}e_{Uf})$	**ERROR:** La llave especial no es la esperada por que los datos han sido modificados y no puede descifrar a la llave pública.

Caso 3. Uso de las llaves públicas y privadas originales.

Datos originales: e_{Uf}, e_R, $d_{Uf.}$

Datos modificados: s, r.

Generación:

$*s = d_{Uf}(*r)$	Se firman los resultados modificados
$*c = k_U(*r)$	Se cifran los resultados.
$\hat{}d_{Uf} = k_U(d_{Uf})$	Se cifra la llave privada.
$k_{ESP} = *s$	Se crea la llave especial.

$^e_{Uf} = k_{ESP}(e_{Uf})$	Se cifra la llave pública.
$p = e_R(k_U)$	Se cifra la llave simétrica.
Transporte (*c, $^d_{Uf}$, $^e_{Uf}$, *p)	
Verificación:	
$k_U = d_R$ (p)	La llave pública corresponde con la privada.
$*r = k_U^{-1}(c)$	Los resultados se descifran de manera correcta.
$d_{Uf} = k_U^{-1}(^d_{Uf})$	La llave privada se descifra de manera correcta.
$s = d_{Uf}$ (r_T)	Se capturan los resultados del acta y se firman.
$k_{ESP} = s$	Se crea la llave especial.
$e_{Uf} \neq k_{ESP}(^e_{Uf})$	**ERROR:** La llave especial no es la esperada por que los datos han sido modificados y no puede descifrar a la llave pública.

Caso 4. Uso de las llaves pública, privada y firma originales, solo se modifican los resultados.

Datos originales: e_{Uf}, e_R, d_{Uf}, s.

Datos modificados: r.

Generación:

$*s = d_{Uf}$ (*r)	Se firman los resultados modificados.
$*c = k_U(*r)$	Se cifran los resultados.
$^d_{Uf} = k(d_{Uf})$	Se cifra la llave privada.
$k_{ESP} = s$	Se crea la llave especial.
$^e_{Uf} = k_{ESP}(e_{Uf})$	Se cifra la llave pública.
$p = e_R(k_U)$	Se cifra la llave simétrica.
Transporte (*c, $^d_{Uf}$, $^e_{Uf}$, *p)	
Verificación:	
$k_U = d_R$ (p)	La llave pública corresponde con la privada.
$*r = k_U^{-1}(c)$	Los resultados se descifran de manera correcta.
$d_{Uf} = k_U^{-1}(^d_{Uf})$	La llave privada se descifra de manera correcta.
$s = d_{Uf}$ (r_T)	Se leen los resultados del acta y se firman.
$k_{ESP} = s$	Se crea la llave especial.
$e_{Uf} = k_{ESP}(^e_{Uf})$	La llave especial es la esperada y descifra a la llave pública.
e_{Uf}(*r, s)	**ERROR:** La verificación falla por que la firma no corresponde con los resultados modificados.

Caso 5. No se modifican los datos y se usan las llaves originales.

Datos originales: s, r, e_{Uf}, e_R, d_{Uf}.

Datos modificados:---

Generación:

$s = d_{Uf}$ (r)	Se firman los resultados.
$c = k_U(r)$	Se cifran los resultados.

$^d_{Uf} = k(d_{Uf})$	Se cifra la llave privada.
$k_{ESP} = s$	Se crea la llave especial.
$^e_{Uf} = k_{ESP}(e_{Uf})$	Se cifra la llave pública.
$p = e_R(k_U)$	Se cifra la llave simétrica.

Transporte (c, $^d_{Uf}$, $^e_{Uf}$, p)

Verificación:

$k_U = d_R\ (p)$	La llave pública corresponde con la privada.
$r = k_U^{-1}(c)$	Los resultados se descifran de manera correcta.
$d_{Uf} = k_U^{-1}(^d_{Uf})$	La llave privada se descifra de manera correcta.
$s = d_{Uf}\ (r_T)$	Se leen los resultados del acta y se firman.
$k_{ESP} = s$	Se crea la llave especial.
$e_{Uf} = k_{ESP}(^e_{Uf})$	La llave especial es la esperada y descifra la llave pública.
$e_{Uf}(r, s)$	**ÉXITO:** La verificación es correcta.

Opción 2. Envío de la firma a través de un canal de comunicaciones seguro.

Los resultados de las pruebas para el envío de datos de la urna electrónica al equipo analizador de resultados utilizando la opción de enviar la firma digital a través de un canal de comunicaciones seguro se muestran en la Tabla 8.

Caso	Datos originales	Datos modificados	Resultado de la verificación
1	---	e_{Uf}, e_R, d_{Uf}, s, r	ERROR
2	e_{Uf}, e_R	s, r, d_{Uf}	ERROR
3	e_{Uf}, e_R, d_{Uf}	s, r	ERROR
4	e_{Uf}, e_R, d_{Uf}, s	r	ERROR
5	e_{Uf}, e_R, d_{Uf}, s, r	---	ÉXITO

Tabla 8. Resultados para la opción 2 de la urna al equipo analizador de resultados.

Caso 1. Suplantación de todas las llaves y datos.

Datos originales: ---

Datos modificados: s, r, d_{Uf}, e_{Uf}, e_R.

Generación:

$*s = *d_{Uf}\ (*r)$	El atacante crea sus propios resultados y los firma con su llave privada.
$*c = k_U(*r)$	Se crea la llave simétrica y se cifran los resultados.
$k_{ESP} = *s$	Se crea la llave especial con la firma modificada.
$^e_{Uf} = k_{ESP}(*e_{Uf})$	Se cifra la llave pública creada por el atacante con la llave especial.
$p = *e_R(k_U)$	Se cifra la llave simétrica de resultados con la llave pública creada por el atacante.

Transporte (*c, ^e_{Uf}, *p)

Verificación:

$k_U \neq d_R$ (p).	**ERROR:** La llave pública no corresponde con la privada.

Caso 2.Uso de las llaves públicas originales, el resto de los datos se modifica.

Datos originales: e_{Uf}, e_R

Datos modificados: s, r,$d_{Uf.}$.

Generación:

*s = *d_{Uf} (*r)	El atacante crea sus propios resultados y los firma con su llave privada.
*c = k_U(*r)	Se crea la llave simétrica y se cifran los resultados.
k_{ESP} = *s	Se crea la llave especial con la firma modificada.
^e_{Uf} = k_{ESP}(e_{Uf})	Se cifra la llave pública original con la llave especial.
p = e_R(k_U)	Se cifra la llave simétrica de resultados con la llave pública original.

Transporte (*c, ^e_{Uf}, *p)

Verificación:

$k_U = d_R$ (p)	La llave pública corresponde con la privada.
*r = k_U^{-1}(c)	Los resultados se descifran de manera correcta.
s	Se recupera la firma del canal de comunicaciones seguro.
k_{ESP} = s	Se crea la llave especial.
$e_{Uf} \neq k_{ESP}$(^e_{Uf})	**ERROR:** La llave especial no es la esperada por que la firma original no es igual a la firma modificada y no puede descifrar a la llave pública.

Caso 3. Uso de las llaves públicas y privadas originales, el resto de los datos se modifica.

Datos originales: e_{Uf}, e_R, $d_{Uf.}$

Datos modificados: s, r

Generación:

*s = d_{Uf} (*r)	El atacante crea sus propios resultados y los firma con la llave privada original.
*c = k_U(*r)	Se crea la llave simétrica y se cifran los resultados.
k_{ESP} = *s	Se crea la llave especial con la firma modificada.
^e_{Uf} = k_{ESP}(e_{Uf})	Se cifra la llave pública original con la llave especial.
p = e_R(k_U)	Se cifra la llave simétrica de resultados con la llave pública original.

Transporte (*c, ^e_{Uf}, *p)

Verificación:

$k_U = d_R$ (p)	La llave pública corresponde con la privada.

$*r = k_U^{-1}(c)$	Los resultados se descifran de manera correcta.
s	Se recupera la firma del canal de comunicaciones seguro.
$k_{ESP} = s$	Se crea la llave especial.
$e_{Uf} \neq k_{ESP}(\hat{}e_{Uf})$	**ERROR:** La llave especial no es la esperada por que la firma original no es igual a la firma modificada y no puede descifrar a la llave pública.

Caso 4. Uso de las llaves pública, privadas y de la firma originales, solo se modifican los resultados.

Datos originales: e_{Uf}, e_R, d_{Uf}, s

Datos modificados: r

Generación:

$*s = d_{Uf}(*r)$	El atacante crea sus propios resultados y los firma con la llave privada original.
$*c = k_U(*r)$	Se crea la llave simétrica y se cifran los resultados.
$k_{ESP} = s$	Se crea la llave especial con la firma original.
$\hat{}e_{Uf} = k_{ESP}(e_{Uf})$	Se cifra la llave pública original con la llave especial.
$p = e_R(k_U)$	Se cifra la llave simétrica de resultados con la llave pública original.

Transporte ($*c$, $\hat{}e_{Uf}$, $*p$)

Verificación:

$k_U = d_R(p)$	La llave pública corresponde con la privada.
$*r = k_U^{-1}(c)$	Los resultados se descifran de manera correcta.
s	Se recupera la firma del canal de comunicaciones seguro.
$k_{ESP} = s$	Se crea la llave especial.
$e_{Uf} = k_{ESP}(\hat{}e_{Uf})$	La llave especial es la esperada y descifra a la llave pública.
$e_{Uf}(*r, s)$	**ERROR:** La verificación falla por que la firma original no corresponde con los datos modificados

Caso 5. No se modifican los resultados y se utilizan las llaves originales.

Datos originales: s, r, e_{Uf}, e_R, d_{Uf}.

Datos modificados:---

Generación:

$s = d_{Uf}(r)$	Se generan los resultados y se firman con la llave privada original.
$c = k_U(r)$	Se crea la llave simétrica y se cifran los resultados.
$k_{ESP} = s$	Se crea la llave especial con la firma original.
$\hat{}e_{Uf} = k_{ESP}(e_{Uf})$	Se cifra la llave pública original con la llave especial.
$p = e_R(k_U)$	Se cifra la llave simétrica de resultados con la llave pública original.

Transporte (c, ^d_{Uf}, ^e_{Uf}, p)

Verificación:

$k_U = d_R$ (p)	La llave pública corresponde con la privada.
$r = k_U^{-1}$(c)	Los resultados se descifran de manera correcta.
s	Se recupera la firma del canal de comunicaciones seguro.
$k_{ESP} = s$	Se crea la llave especial.
$e_{Uf} = k_{ESP}$(^e_{Uf})	La llave especial es la esperada y descifra la llave pública.
e_{Uf}(r, s)	**ÉXITO:** La verificación es correcta.

Los protocolos diseñados son muy eficientes ya que detectan si los archivos de configuración o de resultados han sido modificados sin importar que el atacante tenga acceso a las llaves públicas o privadas. El crear una llave especial formada de la firma de los datos y el manejo que se da de esta misma es una muy buena medida de seguridad que garantiza que al ser modificados la llave no será correcta y la verificación fallará además el hecho de que ésta llave no se almacene en ningún equipo ni se transporte por ningún canal de comunicaciones inseguro hace que sea imposible el ataque de suplantación.

En el caso de los procedimientos realizados en la urna electrónica, al detectarse un error se muestra la ventana que indica que la validación ha sido incorrecta, se reporta el error en la bitácora y el sistema se apaga. En el equipo que analiza los resultados, solo se cuentan con los programas para descifrar y firmar de manera independiente por lo que solo se muestra el aviso de “La verificación no es correcta”.

4.3 PRUEBAS A LA ARQUITECTURA.

Después de la revisión de los trabajos relacionados y de analizar las propiedades que un sistema de voto electrónico debe poseer para ser considerado seguro y auditable se analizó la manera en que éstas son cubiertas con los componentes de auditoría, de seguridad y protocolos criptográficos diseñados y agrupados en la arquitectura, los resultados son los siguientes:

Anónimo, privado y no coaccionable.
Con el almacenamiento aleatorio del voto, se evita una relación voto – votante, además el método de activación del equipo que es un código de barras contiene números generados de manera aleatoria que no contienen ningún tipo de información que pudiera relacionarse con el votante. Además en la bitácora se registra el momento en que se emitió un voto pero sólo indicando en que elección fue, no se tiene una hora que indique cuando se votó por alguna opción. El usuario no tiene manera alguna de poder comprobar a otras personas que votó por alguien en particular ya que el único acceso que tiene al comprobante impreso es visual.

Elegible, auténtico y democrático. Esta propiedad no se trabajó en el proyecto ya que no se incluyó el método de activación del equipo, sólo se menciona como un elemento más de la arquitectura a implementar haciendo la indicación que un usuario no podrá votar en más de una ocasión.

Íntegro. Los votos se encuentran protegidos dentro del equipo y no hay manera de acceder a ellos a través de un dispositivo externo como podría ser un teclado, en cuanto a la seguridad cada voto se cifra de manera individual con su propia llave, el comprobante electrónico se cifra con una llave simétrica de 128 bits que se genera en el momento que se requiere, al finalizar la jornada electoral esta llave se cifra con la llave pública del equipo que analiza los resultados por lo que los *EBI* sólo se podrán descifrar en el lugar donde se encuentre esta llave.

Exacto y verificable. El archivo de votos se guarda en distintos medios de almacenamiento de una manera que garantiza que siempre contendrán el mismo número de votos, además se cuenta con el comprobante electrónico o EBI y con el comprobante impreso en caso de que se requiera algún recuento, todos los eventos que se vayan realizando en el equipo quedan almacenados en un archivo de eventos o bitácora.

Confiable, fácil de utilizar y verificable por el individuo. Esta propiedad se probó como un resultado de la aplicación del esquema de pruebas paralelas, demostrando que un alto porcentaje de los usuarios consideran al sistema confiable, en cuanto al uso, la gran mayoría de los usuarios opinó que el uso del sistema es entre muy simple y simple. La verificación individual se cubre parcialmente con el reporte directo del voto y de una manera total con el comprobante impreso.

La Tabla 9 muestra la manera en que la arquitectura cubre éstas propiedades.

Propiedad	Cubierta
Anónimo	COMPLETAMENTE
Privado	COMPLETAMENTE
No coaccionable	COMPLETAMENTE
Elegible	PARCIALMENTE
Auténtico	PARCIALMENTE
Democrático	PARCIALMENTE
Íntegro	COMPLETAMENTE
Exacto	COMPLETAMENTE
Verificable	COMPLETAMENTE
Confiable	COMPLETAMENTE
Fácil de utilizar	COMPLETAMENTE
Verificable por el individuo	COMPLETAMENTE

Tabla 9. Propiedades cubiertas por la arquitectura.

4.4 PRUEBAS AL CÓDIGO FUENTE.

4.4.1 Pruebas con Flawfinder.

Flawfinder es un programa escrito en *Phyton* que examina el código fuente y reporta posibles vulnerabilidades que son registradas de acuerdo a su nivel de riesgo, ésta aplicación es de código abierto y funciona en los sistemas basados en *Unix*.

Instalación.

Descargar el archivo binario (RPM) o el archivo que contiene el código para compilar, la dirección se encuentra en la parte de anexos, si se descarga el archivo binario la instalación es la siguiente:

```
rpm -Uvh flawfinder-*.noarch.rpm
```

Si se descarga el archivo tar la instalación debe realizarse de la siguiente manera:

```
gunzip  flawfinder-*.tar.gz
tar xvf flawfinder-*.tar
cd flawfinder-*
su
make install
```

Ejecución.

Una vez instalado el programa, para obtener los resultados del análisis del código fuente que se desean se debe ejecutar el siguiente comando:

```
flawfinder –inputs Directorio_de_trabajo –quiet –html > nombreSalida.html
```

En el caso del proyecto actual, los resultados se obtuvieron de la siguiente manera:

```
vi resultadosFlawFinder.html
flawfinder –inputs urna –quiet –html > resultadosFlawFinder.html
```

Los resultados que arrojó *Flawfinder* están relacionados con el sobre flujo especialmente cuando se reserva memoria para almacenar cadenas de caracteres, recomendando que uno esté seguro que reservó el suficiente espacio, otra recomendación es que al utilizar el operador "%s" hay que asegurarse que se tiene un límite suficiente para no generar un sobre flujo.

Los resultados son:

[0+] 20 [1+] 14 [2+] 2 [3+] 2 [4+] 2 [5+] 0, dónde el número entre corchetes representa el nivel de peligro, 0 más bajo, 5 más alto y el otro número la cantidad de riesgos potenciales que se encontraron, hay que señalar que ésto no representa que el programa este mal o no vaya a funcionar, solo da recomendaciones de donde se deben revisar algunos aspectos para evitar problemas. Los

resultados, que se pueden observar en el anexo C, son satisfactorios ya que se tienen pocas posibles vulnerabilidades de alto nivel.

4.4.2 Pruebas con RATS.

RATS es una herramienta de código abierto utilizada para la revisión de programas en lenguaje *C*, *C++* y *Perl* entre otros que localiza diversos errores en el código fuente tales como sobre flujos, calidad de la generación de números aleatorios y que proporciona una lista de los posibles problemas contenidos en el código y una asesoría para resolverlos.

Instalación.

Después de descargar el archivo, la instalación debe realizarse de la siguiente manera:

```
tar –zxf rats-2.1.tar.gz
cd rats-2.1
./configure
make
make install
```

Ejecución.

Una vez instalado el programa, la manera de ejecutarlo es la siguiente:

```
rats –d nombre.cpp –quiet –html > nombreSalida.html
```

Para el proyecto se realizó lo siguente:

```
cd urna
vi resultadosRATS.html
rats –d *.cpp –quiet –html > resultadosRATS.html
```

Los resultados que *RATS* generó están relacionados con el sobre flujo al momento de reservar espacio para cadenas de caracteres. Cuando se utilice el comando "*sprintf*" hay que asegurarse que alguno de los parámetros no contenga caracteres que no puedan ser manejados además hace una sugerencia sobre la generación de números aleatorios indicando que "*srand*" no es muy seguro para la generación de éstos mismos, pero al utilizarse sólo para desordenar un conjunto de números ordenados no representa algún peligro, el manejo aleatorio de las llaves se realiza con funciones de *OpenSSL* que tienen una buena entropía (medida que indica que tan aleatorios son los datos generados). Estos resultados son:

- Severidad alta relacionada con el tamaño del buffer.
  Detectadas: 45.
- Severidad alta relacionada con el uso de *sprintf*
  Detectadas: 15

- Severidad alta relacionada con *strcpy*.
  Detectadas: 1.
- Severidad media relacionada con *read*.
  Detectadas: 12.
- Severidad media relacionada con EVP_DecryptUpdate.
  Detectadas: 2
- Severidad media relacionada con EVP_EncryptUpdate.
  Detectadas: 3
- Severidad media relacionada con *srand*.
  Detectadas: 1

Estas son advertencias de posibles vulnerabilidades, revisando el código se aseguró que los tamaños reservados son suficientes por lo que los resultados, que se pueden revisar en el anexo D, son satisfactorios.

4.4.3 Documentación del código fuente.

Se utilizó el programa *Doxygen* que es un sistema de documentación para lenguajes como *C*, *C++* o *Java* entre otros y que genera manuales en formato *html*, *PDF*, *Latex* ó estilo *Linux*. La aplicación de acceso libre está desarrollada bajo *Linux* y *Mac OS X* pudiendo ser transportada a otros sistemas como *Windows*.

Instalación.

El programa viene incluido en el sistema operativo *Linux* o puede descargarse de la página de *Doxygen*.

Ejecución.

Primero de debe generar un archivo de configuración:

```
doxygen –g archivoConfiguracion
```

Posteriormente se debe editar este archivo para adecuarlo las necesidades y gustos del usuario, en la edición se deben llenar ciertos campos con la información necesaria y cambiar parámetros de *YES* a *NO* según lo que se desee. Existe una interfaz gráfica para realizar la configuración, ésta se carga a través del comando:

```
Doxywizard
```

Una vez finalizada la edición se ejecuta el comando:

```
doxygen archivoConfiguracion.
```

CAPÍTULO 5

CONCLUSIONES Y TRABAJO FUTURO.

En este capítulo se presentan las conclusiones a las que se llegaron al finalizar el desarrollo del proyecto, cubriendo la parte de auditoría, seguridad, arquitectura y la manera de trabajar con un sistema de voto electrónico. También se mencionan los trabajos futuros que se pueden desarrollar a partir de éste proyecto.

5.1 CONCLUSIONES

Auditoría.

El trabajar con el concepto de redundancia como base garantiza que se tengan diversas versiones de un mismo elemento, ésto hace al sistema más confiable además de permitir detectar si se han cometido alteraciones durante la votación gracias a elementos como las auditorías intermedias y los comprobantes almacenados en diversos formatos y medios. Los elementos a incluir deben ser de acuerdo a las necesidades en cada etapa, de acuerdo a esto, la arquitectura de auditoría diseñada es eficiente ya que permite que en la etapa de pre votación se tengan elementos que permiten comprobar el correcto funcionamiento del equipo y de la configuración del sistema, para la etapa de votación tener elementos que incrementen la confianza de los votantes al momento en que están utilizando el sistema además de contar con medios para verificar que todo el proceso se está llevando a cabo de manera adecuada, finalmente para la etapa de post votación se tienen suficientes elementos que contienen los resultados finales en distintos formatos lo que permite poder realizar una comparación de los resultados registrados en cada uno de ellos, además de permitir un recuento de los votos ya sea de manera manual utilizando los *VVPAT* o en otro equipo usando los *EBI*. En el caso de la bitácora, éste documento contiene información sobre todos los eventos que ocurren en el equipo, la versión que se desarrolló además permite realizar un recuento sobre el total de votos registrados en cada elección.

Sobre el funcionamiento general del sistema, los resultados que arrojó la encuesta realizada a los usuarios muestran que el uso del equipo es sencillo, que la gran mayoría confía en él y que los elementos a los que tienen acceso como son el comprobante impreso y el reporte de la contribución del voto hacen que su confianza en el sistema aumente, sobre estos elementos es importante hacer notar la necesidad de una capacitación hacia el votante antes de su participación, de ésta manera el porcentaje de usuarios que no notó el comprobante impreso y en especial el incremento de votos disminuirá y muy posiblemente aumentará el porcentaje del incremento de confianza en el sistema. El hecho de que conocer como está realizado el sistema genere mayor confianza refuerza la propuesta de que el código fuente de éstos sistemas debe ser de acceso público y por lo tanto debe contar con una adecuada documentación que explique qué es lo que realiza cada una de sus distintas funciones o clases. Las aplicaciones utilizadas para verificar el código arrojan resultados que permiten revisar puntos específicos acerca de las posibles vulnerabilidades, pero siempre con la indicación de que estos programas no deben suplantar a una revisión detallada. El esquema de pruebas paralelas demostró ser eficiente para verificar el funcionamiento del sistema en general y para comprobar la veracidad de los resultados que éste arroja, proporcionando además una buena opción para obtener las opiniones de los usuarios sobre él. Es importante hacer notar que aunque se tiene una gran cantidad de elementos para comprobar el funcionamiento del equipo, el proceso de votación y los resultados finales ninguno de ellos viola alguna de las propiedades que los sistemas de voto electrónico deben contener.

Seguridad.

Si bien los votos son el elemento fundamental de estos sistemas, existen otros elementos que se deben proteger que son los archivos críticos que incluyen a los archivos de configuración y de resultados que en este tipo de sistemas son los que corren mayor riesgo al ser transportados por canales de comunicación considerados inseguros. En cuanto al manejo de las llaves, el generar las simétricas en el momento que se requieren utilizando métodos que tienen un buen nivel de entropía que se traduce en una alta aleatoriedad y con un adecuado tamaño de llave para ambos tipos de cifrado garantiza que el sistema será seguro por mucho tiempo. Los protocolos criptográficos que se diseñaron demostraron ser eficientes al no depender de la seguridad de ninguna de las llaves privadas utilizadas, el tomar como última medida de verificación la integridad de los datos creados originalmente permite detectar una modificación a los datos sin importar que el atacante cree su propio conjunto de llaves y archivos o que llegue a tener acceso a los originales. El instalar con anterioridad las llaves en los distintos equipos las protege contra el ataque de suplantación, ésta instalación debe seguir ciertas normas para realizarse como puede ser el que todos los involucrados en la elección estén presentes para evitar que las llaves sean alteradas al momento de su instalación.

Sobre el desarrollo, utilizar *OpenSSL* permite generar primitivas criptográficas adecuadas ya que cuenta con algoritmos que han sido probados por expertos y que son considerados muy seguros, teniendo eso como base la construcción de los protocolos que es dónde se presentan la mayor cantidad de ataques se realizó sobre elementos seguros lo que permitió el desarrollo de los protocolos sin la necesidad de preocuparse por las primitivas. Gracias al tipo de implementación que se tiene con *OpenSSL* basada en el uso de librerías y llamadas a funciones que sólo deben llenarse con los datos requeridos, hace que el realizar cambios de algoritmo de cifrado sea muy sencillo teniendo en la mayoría de las ocasiones que cambiar solamente el nombre del algoritmo o la función que se utiliza.

Arquitectura.

La arquitectura diseñada cumple satisfactoriamente con el objetivo principal para el que fue creada, que era resolver el problema de la plataforma segura ya que cubre los puntos vulnerables de éste tipo de sistemas desde el punto de vista de la seguridad y de la auditoría especificando claramente los elementos que se deben encontrar en cada una de las diferentes capas que la conforman y la interacción que debe existir entre ellos, lo que permite realizar el diseño e implementación de sistemas de voto electrónico que generen altos niveles de confianza en el público que los utiliza. La arquitectura cubre completamente los puntos vulnerables tanto de seguridad como de auditoría y los que se cubren parcialmente son debido a que no fueron tratados durante el desarrollo del proyecto.

Trabajo con el sistema.

Si bien se puede pensar que las tres etapas de un sistema de voto electrónico están muy relacionadas entre si, separarlas permite trabajar de una manera modular lo que facilita el manejo de un sistema

que en conjunto podría resultar muy complejo, la único que se debe considerar en la relación entre las etapas son los archivos que entrega una de ellas como salida y que recibe otra como entrada así como las restricciones que se deben tener y las condiciones que se deben cumplir para poder pasar de una etapa a la otra.

5.2 TRABAJO FUTURO.

Se puede utilizar la arquitectura diseñada como base para la creación de una que cubra las etapas de generación de archivos y de análisis de resultados. Si bien el protocolo que se diseñó cubre la seguridad y la integridad de los datos que se obtienen de la etapa de generación de archivos no se trabajan los elementos de seguridad y auditoría que ésta debe poseer al momento de su creación, de manera similar con la etapa de análisis de resultados, la arquitectura diseñada maneja la seguridad de los datos que recibe, pero se debe crear una arquitectura más específica para cuando se está realizando el análisis de los mismos.

Dentro de lo que fue la etapa de votación no se trató lo referente a la activación del equipo, la identificación y la validación de usuarios, éste es otro punto sobre el que se puede trabajar, revisando como se activa el equipo, si existe otro sistema que lo active, cómo tener una comunicación segura entre ellos de manera que no haya otro equipo que pueda activarlo, el manejo que se le debe dar a la base de datos que contenga los usuarios válidos, qué información contenida en ésta misma debe estar protegida y evitar que sea modificada.

Como trabajos futuros para el proyecto desarrollado se tiene el incremento de la calidad del código fuente, que consiste en reducir el número de advertencias que se generan al analizarlo, desarrollar programas y configurar la impresora para permitir la generación de códigos de barra, para el control de la hora de encendido del equipo se debe eliminar la dependencia del reloj del sistema teniendo como opción tomar la hora de algún dispositivo externo como un reloj o un GPS, hacer que el sistema sea totalmente configurable a partir de información contenida en archivos de texto y trabajar con lo que es la seguridad propia del equipo donde se realiza la votación, que incluye seguridad física, del sistema operativo y de la aplicación.

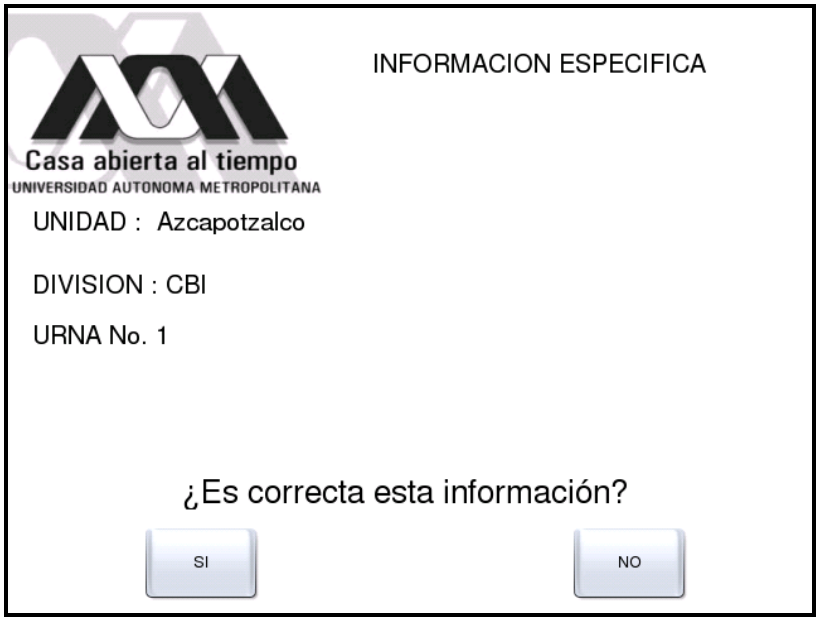
Finalmente como aportes a sistemas que no son de voto electrónico, es conveniente que los conceptos que se manejan se apliquen a otro tipo de aplicaciones para que éstas cuenten con elementos que permitan verificar que su funcionamiento y los resultados que arrojan son adecuados, por ejemplo en una aplicación bancaria se debe tener una manera de demostrar que las operaciones fueron realizadas de manera adecuada o en algún programa de análisis o procesamiento de datos se debe poder verificar que los resultados son congruentes con lo esperado, esto además de incluir las recomendaciones sobre la calidad del código fuente.

ANEXOS.

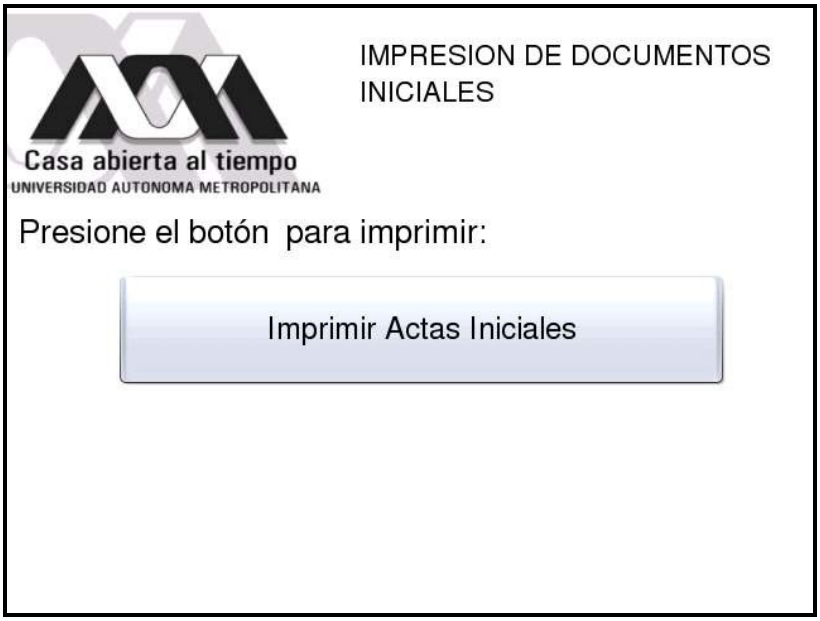
ANEXO A. CAPTURAS DE PANTALLAS.

Captura 1. Pantalla inicial. *Muestra los pasos a seguir durante la configuración del equipo.*

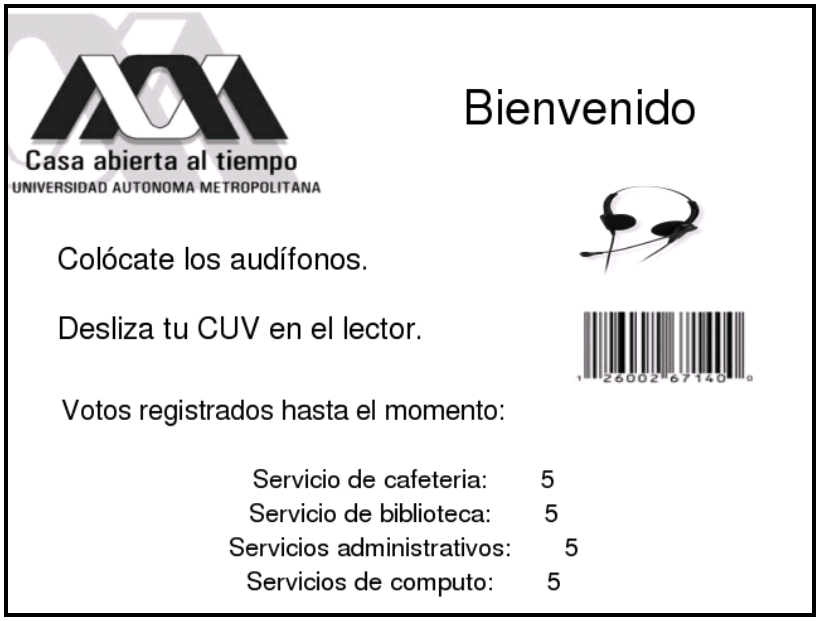
Captura 2. Información de los componentes. *Reporta el estado del hardware.*



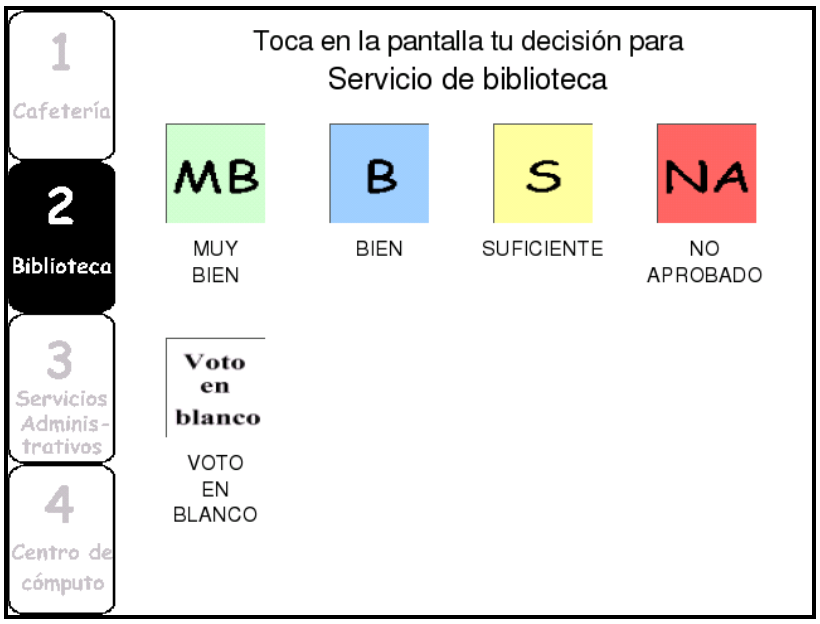
Captura 3. Información de la configuración. *Muestra los datos específicos del equipo.*



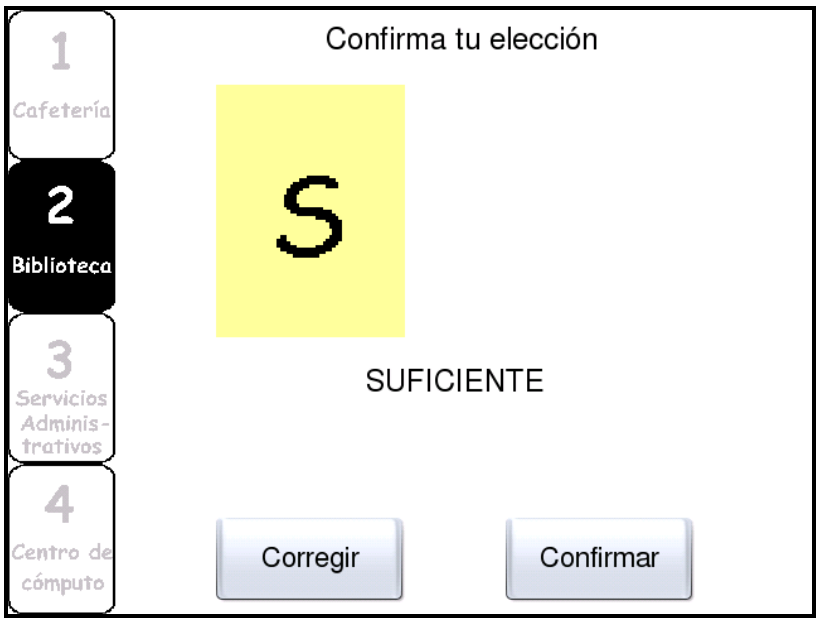
Captura 4. Impresión de documentos iniciales. *Pantalla para la impresión de actas iniciales, de apertura y de comprobante de componentes.*



Captura 5. Bienvenida al usuario. *Pantalla que el usuario observa, se muestran los votos que se han registrado hasta el momento.*

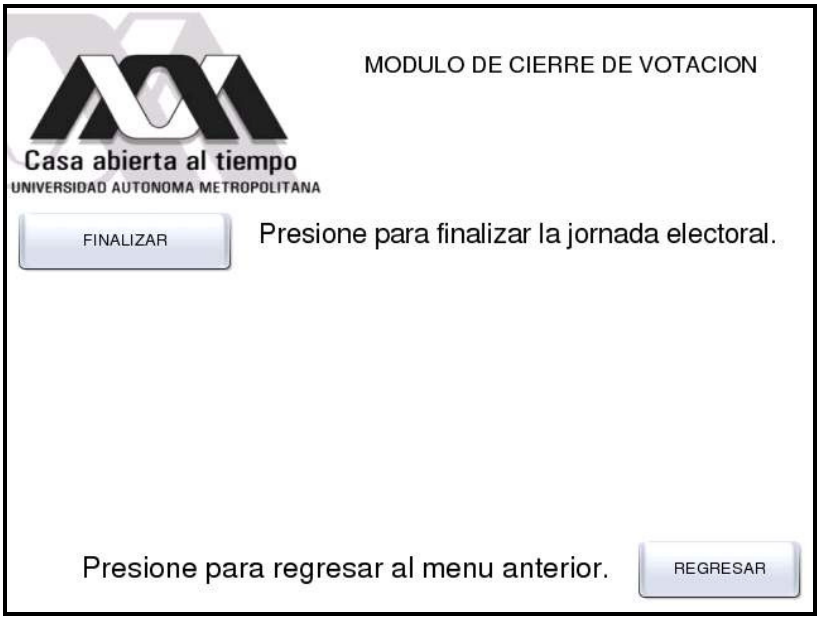


Captura 6. Boleta. *Muestra las diferentes elecciones y las opciones que se tienen en cada una.*

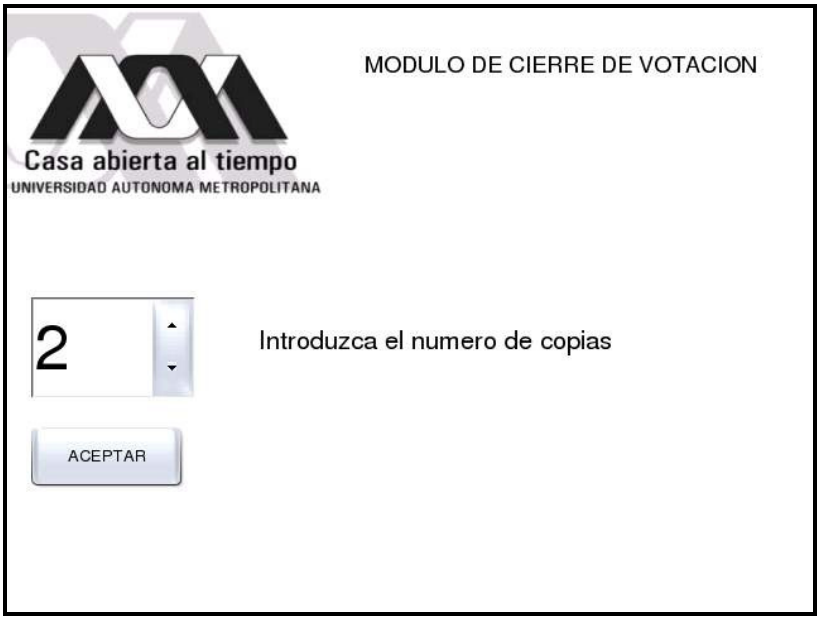


Captura 7. Confirmar o corregir. *Aquí el usuario puede corregir o confirmar su voto.*

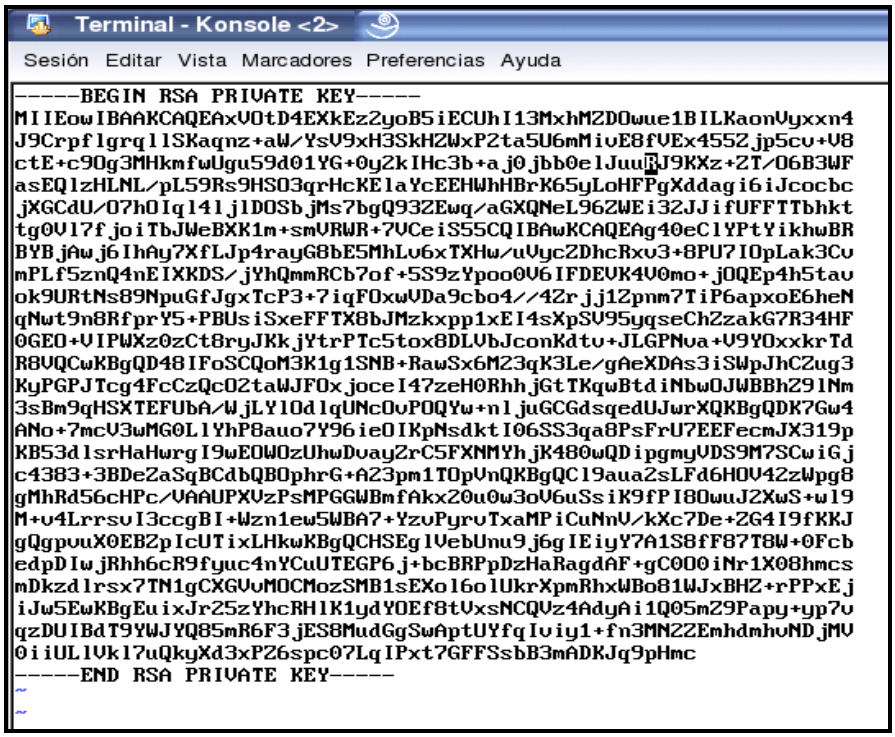
Captura 8. Reporte de contribución del voto. *Pantalla de despedida, el número de votos se habrá incrementado en uno después de la participación del usuario.*



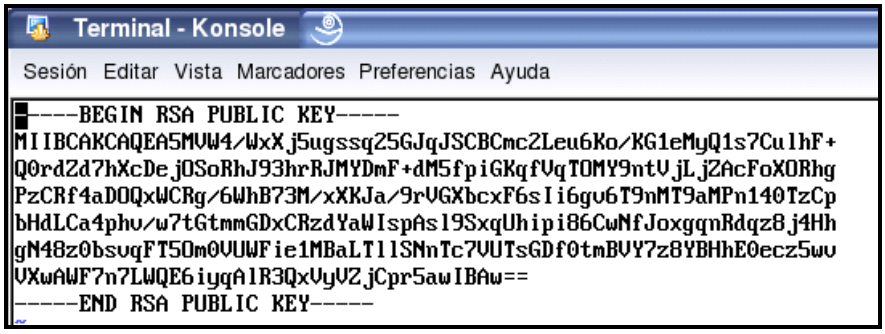
Captura 9. Menú del administrador. *Pantalla que utilizan los funcionarios para la finalización de la jornada electoral, tiene la opción de regresar a la etapa de votación.*



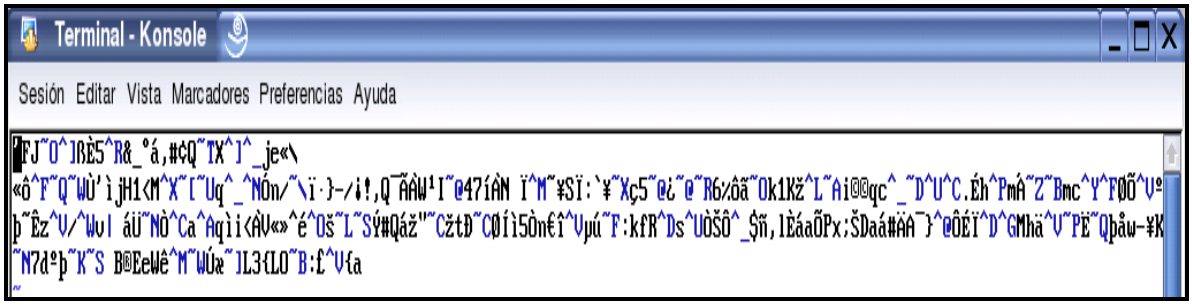
Captura 10. Impresión de documentos finales. *Pantalla para imprimir las actas iniciales con determinado número de copias.*



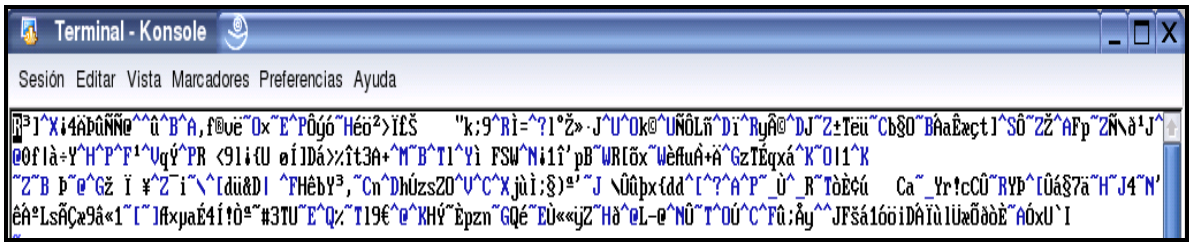
Captura 11. Llave privada en formato PEM.



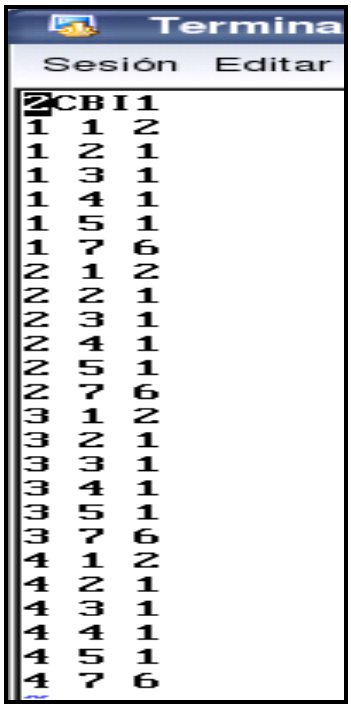
Captura 12. Llave pública en formato PEM



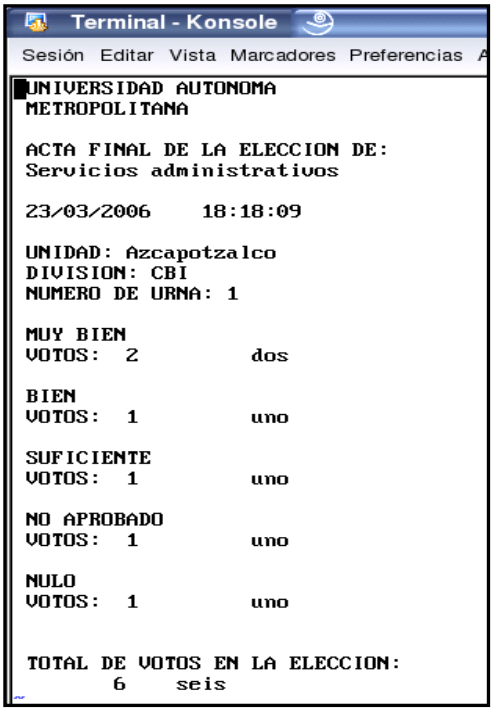
Captura 13. Firma digital.



Captura 14. Documento cifrado.



Captura 15. Archivo de resultados



Captura 16. Acta final

ANEXO B. VERSIONES DEL CÓDIGO FUENTE.

En esta parte se describen brevemente las distintas versiones del código que se desarrollaron con el objetivo de llevar un control adecuado de los cambios que se han realizado y por ser una buena práctica en cuánto a la auditoría del código fuente.

05-11-19-AntesSimétrico.
Esta es la versión original del código fuente que se generó en el desarrollo del proyecto "Tinochtín".

05-11-19-EncriptadoSimétrico.
Primera versión con el cifrado simétrico, en esta versión solo se cuenta con el cifrado y los datos cifrados se almacenan en un archivo de texto.

Archivos añadidos:
Clave.h
GeneraLlaves.cpp
GeneraLlaves.h
Simetrico.cpp
Simetrico.h

Archivos modificados:
FrmImpresiónInicial.ui.h
MainWindow.cpp
Urna.h
Voto.cpp

05-12-26-ActasArregladasEBI.
Aquí se modificaron los programas de generación de actas para adecuarlos a los datos que se tendrían en una elección realizada en la UAM, también se modificó el comprobante electrónico para que su análisis sea más sencillo cambiando el nombre de la elección y el nombre de la opción elegida por su equivalente en clave numérica, dejando el nombre de la opción y de la elección solo para el comprobante impreso.

Archivos añadidos:
En esta versión no se añadió ningún archivo.

Archivos modificados:
Casilla.cpp
Casilla.h
FrmMostrarInformacion.ui.h
FrmFinalizarUrna.ui.h
Totaliza.cpp
Urna.cpp
Voto.cpp

06-01-16-Directorios.
Se modificaron los directorios en los que se encuentran las imágenes y los sonidos que se presentan durante la ejecución del programa.

Archivos añadidos:
En esta versión no se añadió ningún archivo.

Archivos modificados:
FrmBienvenida.ui.h
FrmMsgGracias.ui.h
FrmVerificaSonido.ui.h

06-01-30-Limpieza.
Se depuró el código que se tenía hasta el momento eliminando las rutinas comentadas y algunas que ya no eran necesarias como las que incluían ciertas llamadas al sistema sobre todo las relacionadas con el cifrado y descifrado simétrico.

Archivos añadidos:
En esta versión no se añadió ningún archivo.

Archivos modificados:
Boleta.cpp
Casilla.cpp
CtlOpcion.cpp
Cuv.cppFrm
Bienvenida.ui.h
FrmMsgGracias.cpp
Main.cpp
MainWindow.cpp
Totaliza.cpp
Voto.cpp

06-02-04-NuevoEncriptadoSimetrico.
Nueva versión del código con el cifrado simétrico, se modificó la forma en que se almacenan los datos cifrados, haciéndose ahora en un archivo binario de acceso secuencial, además se utiliza una llave distinta para cada uno de los documentos que se generan como las actas iniciales, actas finales, acta de apertura, comprobante de componentes y votos, en la versión anterior se tenía una llave para cada acta y para cada voto pero por cuestiones de espacio y facilidad de organización de estas mismas se modificó.

Archivos añadidos:
EncriptadoSimetrico.cpp
EncriptadoSimetrico.h

Archivos modificados:
Clave.h
FrmAdministración.ui.h
FrmConfiguracion.ui.h
FrmImpresionInicial.ui.h
GeneraLlaves.cpp
GeneraLlaves.h
MainWindow.cpp
Urna.h
Voto.cpp

06-02-22-EncriptaYDesencripta.
Por cuestiones de espacio, se modificó el cifrado simétrico, ahora se almacena en un archivo de texto ocupando el archivo cifrado el mismo tamaño en bytes que el archivo de texto simple, antes se cifraba línea por línea escribiendo el resultado en otro archivo, ahora se carga toda la información en un buffer y se cifra por completo, después la información cifrada se almacena en el mismo archivo sobrescribiendo el texto simple. Se agregó también la parte de descifrado con el mismo funcionamiento que el cifrado.

Archivos añadidos:
DesencriptadoSimetrico.cpp
DesencriptadoSimetrico.h

Archivos modificados:
Clave.h
EncriptadoSimetrico.cpp
FrmConfiguracion.ui.h
FrmImpresionInicial.ui.h
FrmInicializarUrna.ui.h
MainWindow.cpp
Totaliza.cpp
Urna.h

06-02-27-InterfazDeLlavePublica.
Se implementó el cifrado y descifrado asimétrico, la firma digital y la verificación de esta misma que incluye además el cálculo del valor *Hash* de los datos como medida para detectar la integridad de los datos, las llaves que se utilizan se representan como números primos, el llenado se realiza con la ayuda de las funciones de la biblioteca BIGNUM de *OpenSSL*.

Archivos añadidos:
DesencriptadoAsimetrico.cpp
DesencriptadoAsimetrico.h
EncriptadoAsimetrico.cpp

EncriptadoAsimetrico.h
FirmaDigital.cpp
FirmaDigital.h
VerificarFirma.cpp
VerificarFirma.h
VerificarIntegridad.cpp
VerificarIntegridad.h

Archivos modificados:
Boleta.cpp
Casilla.cpp
FrmConfiguracion.ui.h
FrmImpresionInicial.ui.h
MainWindow.cpp
Urna.h

06-03-01-ReporteHaciaElVotante.
Se muestra el total de votos registrados en cada elección, no se muestra el total de votos que tiene cada contendiente para que no se conozca el estado de la elección. Esta información se presenta en la pantalla de bienvenida y al finalizar todas las elecciones se muestra en la pantalla de despedida en donde el número de votos en cada elección aumentó en uno.

Archivos añadidos:
ContadoresInternos.h

Archivos modificados:
Boleta.h
Casilla.cpp
FrmBienvenida.ui.h
FrmMsgGracias.cpp
FrmMsgGracias.h
MainWindow.cpp
Urna.h
Voto.cpp

06-03-11-EncriptadoFirmadoCompleto.
Nueva versión del manejo de llave pública y privada, ahora las llaves se almacenan de manera codificada, y con esto se evita el manejo de la biblioteca BIGNUM que requería la creación de llaves cada que se requerían y luego llenar esas llaves con los datos cargados de las llaves originales, ahora basta con cargar los datos en una estructura vacía. Además al tener las llaves codificadas se tiene una mayor seguridad que con su representación en forma de números primos.

Archivos añadidos:
En esta versión no se añadió ningún archivo.

Archivos modificados:
Boleta.cpp
Casilla.cpp

DesencriptadoAsimetrico.cpp
DesencriptadoAsimetrico.h
EncriptadoAsimetrico.cpp
EncriptadoAsimetrico.h
FirmaDigital.cpp
FirmaDigital.h
FrmConfiguracion.ui.h
FrmAdministracion.ui.h
FrmVentanaDeApagado.ui.h
VerificarFirma.cpp
VerificarFirma.h
VerificarIntegridad.cpp
VerificarIntegridad.h

06-03-11-1raVersionConCUV.
El sistema se activa a través de un código de barras y no con el ratón, este cambio se debe a la prueba a la que se someterá el sistema. Además se modificó la ventana de encendido, y se añadió una ventana de bienvenida antes de comenzar con la configuración del equipo.

Archivos añadidos:
FechaEncendido.cpp
FechaEncendido.h
FrmIniciaProceso.ui.h

Archivos modificados:
Cuv.h
FrmBienvenida.ui.h
FrmVentanaApagado.ui.h
FrmFueraDeFecha.ui.h
MainWindow.cpp
MainWindow.h
Urna.h

ANEXO C. RESULTADOS FLAWFINDER.

Flawfinder version 1.26, (C) 2001-2004 David A. Wheeler. Number of dangerous functions in C/C++ ruleset: 158 Examining urna/Totaliza.cpp Examining urna/Bluetooth.h Examining urna/FrmImpresionInicial.ui.h Examining urna/TASiguienteVentana.cpp Examining urna/Casilla.cpp Examining urna/LlavesIndividuales.cpp Examining urna/SleepThread.cpp Examining urna/Simetrico.h Examining urna/SleepThread.h Examining urna/Cuv.h Examining urna/FrmVerificaSonido.ui.h Examining urna/FrmBoleta.ui.h Examining urna/CodigoDeBarraThread.cpp Examining urna/Encriptar.cpp Examining urna/Bluetooth.cpp Examining urna/Imprimir.cpp Examining urna/TAParpadeaPestana.cpp Examining urna/FrmConfigurar.ui.h Examining urna/FechaEncendido.cpp Examining urna/LlavesIndividuales.h Examining urna/MainWindow.h Examining urna/FrmMuestra.ui.h Examining urna/VerificarFirma.h Examining urna/DesencriptadoSimetrico.h Examining urna/TACambiaVentana.h Examining urna/CtlOpcion.cpp Examining urna/Casilla.h Examining urna/FrmInicializarUrna.ui.h Examining urna/Urna.h Examining urna/CtlBoleta.ui.h Examining urna/TareaAsincrona.cpp Examining urna/Voto.h Examining urna/Clave.h Examining urna/TACambiaVentana.cpp Examining urna/FrmMostrarInformacion.ui.h Examining urna/Opcion.h Examining urna/FrmVentanaApagado.ui.h Examining urna/main.cpp Examining urna/BluetoothUrnaThread.h Examining urna/FrmIniciaProceso.ui.h Examining urna/FrmMsgGracias.h Examining urna/FrmConfiguracion.ui.h Examining urna/CtlConfirmar.ui.h Examining urna/SiguienteVentanaEvent.h Examining urna/FrmAdministracionPrevia.ui.h Examining urna/CtlConfirmar.h Examining urna/Urna.cpp Examining urna/MainWindow.cpp Examining urna/DesencriptadoAsimetrico.h Examining urna/Cuv.cpp Examining urna/split2.cpp Examining urna/VerificarIntegridad.h Examining urna/VerificarIntegridad.cpp Examining urna/FrmEspera.ui.h Examining urna/Bitacora.cpp Examining urna/FrmMsgGracias.cpp Examining urna/CodigoDeBarraThread.h Examining urna/Bitacora.h Examining urna/Boleta.h Examining urna/trace.h Examining urna/EncriptadoSimetrico.h Examining urna/FechaEncendido.h Examining urna/CtlOpcion.h Examining urna/Totaliza.h Examining urna/VerificarFirma.cpp Examining urna/FrmFueraDeFecha.ui.h Examining urna/TAMensajeBluetooth.h Examining urna/DesencriptadoAsimetrico.cpp Examining urna/CtlConfirmarBase.ui.h Examining urna/TASiguienteVentana.h Examining urna/Voto.cpp Examining urna/util.h Examining urna/Encriptar.h Examining urna/FirmaDigital.h Examining urna/Imprimir.h Examining urna/Audita.h Examining urna/totaliza.c Examining urna/EncriptadoAsimetrico.h Examining urna/GeneraLlaves.h Examining urna/BluetoothUrnaThread.cpp Examining urna/CtlConfirmar.cpp Examining urna/GeneraLlaves.cpp Examining urna/TareaAsincrona.h Examining urna/FrmAdministracion.ui.h Examining urna/TAParpadeaPestana.h Examining urna/Boleta.cpp Examining urna/EncriptadoAsimetrico.cpp Examining urna/DesencriptadoSimetrico.cpp Examining urna/ContadoresInternos.h Examining urna/FrmReportaError.ui.h Examining urna/EncriptadoSimetrico.cpp Examining urna/FirmaDigital.cpp Examining urna/FrmComponentes.ui.h Examining urna/FrmBienvenida.ui.h Examining urna/TAMensajeBluetooth.cpp Examining urna/Audita.cpp Examining urna/Simetrico.cpp Examining urna/util.cpp Warning: skipping non-regular file --quiet Warning: skipping non-regular file --html urna/totaliza.c:85: [4] (buffer) fscanf: The scanf() family's %s operation, without a limit specification, permits buffer overflows. Specify a limit to %s, or use a different input function. urna/totaliza.c:283: [4] (buffer) fscanf: The scanf() family's %s operation, without a limit specification, permits buffer overflows. Specify a limit to %s, or use a different input function. urna/DesencriptadoAsimetrico.cpp:27: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/DesencriptadoSimetrico.cpp:34: [1] (buffer) read: Check buffer boundaries if used in a loop.

urna/DesencriptadoSimetrico.cpp:42: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/EncriptadoAsimetrico.cpp:26: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/EncriptadoSimetrico.cpp:33: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/EncriptadoSimetrico.cpp:41: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/FirmaDigital.cpp:27: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/Simetrico.cpp:71: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/VerificarFirma.cpp:29: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/Voto.cpp:94: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/Voto.cpp:102: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/Voto.cpp:140: [1] (buffer) read: Check buffer boundaries if used in a loop. urna/Bluetooth.cpp:37: [0] (input) recv: Function accepts input from outside program. Make sure input data is filtered, especially if an attacker could manipulate it. urna/FrmComponentes.ui.h:55: [0] (buffer) fscanf: The scanf() family's %s operation, without a limit specification, permits buffer overflows. Specify a limit to %s, or use a different input function. No risky scanf format detected. urna/FrmComponentes.ui.h:56: [0] (buffer) fscanf: The scanf() family's %s operation, without a limit specification, permits buffer overflows. Specify a limit to %s, or use a different input function. No risky scanf format detected. urna/FrmComponentes.ui.h:66: [0] (buffer) fscanf: The scanf() family's %s operation, without a limit specification, permits buffer overflows. Specify a limit to %s, or use a different input function. No risky scanf format detected. urna/FrmComponentes.ui.h:67: [0] (buffer) fscanf: The scanf() family's %s operation, without a limit specification, permits buffer overflows. Specify a limit to %s, or use a different input function. No risky scanf format detected. urna/totaliza.c:117: [0] (input) fread: Function accepts input from outside program. Make sure input data is filtered, especially if an attacker could manipulate it. Hits = 20 Lines analyzed = 6830 in 1.77 seconds (5390 lines/second) Physical Source Lines of Code (SLOC) = 4658 Hits@level = [0] 6 [1] 12 [2] 0 [3] 0 [4] 2 [5] 0 Hits@level+ = [0+] 20 [1+] 14 [2+] 2 [3+] 2 [4+] 2 [5+] 0 Hits/KSLOC@level+ = [0+] 4.29369 [1+] 3.00558 [2+] 0.429369 [3+] 0.429369 [4+] 0.429369 [5+] 0 Minimum risk level = 0 Not every hit is necessarily a security vulnerability. There may be other security vulnerabilities; review your code!

ANEXO D. RESULTADOS RATS.

Entries in perl database: **33**
Entries in python database: **62**
Entries in c database: **334**
Entries in php database: **55**

RATS results.

Severity: High
Issue: fixed size global buffer
Extra care should be taken to ensure that character arrays that are allocated on the stack are used safely. They are prime targets for buffer overflow attacks.

File: **Bitacora.cpp**
Lines: 26 27 28
File: **Bluetooth.cpp**
Lines: 172
File: **Boleta.cpp**
Lines: 101 156 260
File: **DesencriptadoAsimetrico.cpp**
Lines: 25 73
File: **DesencriptadoSimetrico.cpp**
Lines: 17 18 41 62
File: **EncriptadoAsimetrico.cpp**
Lines: 24 56
File: **EncriptadoSimetrico.cpp**
Lines: 18 19 40 64
File: **FechaEncendido.cpp**
Lines: 30 31
File: **FirmaDigital.cpp**
Lines: 25 62 63 80
File: **FrmMsgGracias.cpp**
Lines: 58
File: **GeneraLlaves.cpp**
Lines: 22 23
File: **Imprimir.cpp**
Lines: 31
File: **LlavesIndividuales.cpp**
Lines: 70
File: **Simetrico.cpp**
Lines: 23 54 55 56
File: **Totaliza.cpp**
Lines: 70 100 137 189 247
File: **VerificarFirma.cpp**
Lines: 27 69
File: **Voto.cpp**
Lines: 144 230 231 232

Severity: High
Issue: sprintf
Check to be sure that the format string passed as argument 2 to this function call does not come from an untrusted source that could have added formatting characters that the code is not prepared to handle. Additionally, the format string could contain `%s' without precision that could result in a buffer overflow.

File: **Bitacora.cpp**
Lines: 43

File: **Casilla.cpp**
Lines: 41
File: **DesencriptadoSimetrico.cpp**
Lines: 31
File: **EncriptadoSimetrico.cpp**
Lines: 30
File: **FirmaDigital.cpp**
Lines: 83
File: **GeneraLlaves.cpp**
Lines: 35
File: **Simetrico.cpp**
Lines: 65 66
File: **Totaliza.cpp**
Lines: 168 270 272 274 276 279 281

Severity: High
Issue: strcpy
Check to be sure that argument 2 passed to this function call will not copy more data than can be handled, resulting in a buffer overflow.

File: **Bluetooth.cpp**
Lines: 28

Severity: Medium
Issue: read
Check buffer boundaries if calling this function in a loop and make sure you are not in danger of writing past the allocated space.

File: **DesencriptadoAsimetrico.cpp**
Lines: 27
File: **DesencriptadoSimetrico.cpp**
Lines: 34 42
File: **EncriptadoAsimetrico.cpp**
Lines: 26
File: **EncriptadoSimetrico.cpp**
Lines: 33 41
File: **FirmaDigital.cpp**
Lines: 27
File: **Simetrico.cpp**
Lines: 71
File: **VerificarFirma.cpp**
Lines: 29
File: **Voto.cpp**
Lines: 94 102 140

Severity: Medium
Issue: EVP_DecryptUpdate
make sure the output buffer is either at least one block less one byte bigger than the input, or that you are sure inputs are always multiples of the block size, and the output buffer is as big as the input.

File: **DesencriptadoSimetrico.cpp**
Lines: 63
File: **Encriptar.cpp**
Lines: 54

Severity: Medium
Issue: EVP_EncryptUpdate
make sure the output buffer is either at least one block less one byte bigger than the input, or that you are sure inputs are always multiples of the block size, and the output buffer is as big as the input.

File: **EncriptadoSimetrico.cpp**
Lines: 65
File: **Encriptar.cpp**
Lines: 41

File: **Simetrico.cpp**
Lines: 98

Severity: Medium
Issue: srand
Standard random number generators should not be used to generate randomness used for security reasons. For security sensitive randomness a crytographic randomness generator that provides sufficient entropy should be used.

File: **Voto.cpp**
Lines: 75

Inputs detected at the following points

Total lines analyzed: **3446**

Total time **0.107096** seconds**32176** lines per second

ANEXO E. EJECUCIÓN EN LA TARJETA BITSYX.

Instalación del entorno de desarrollo.

Compilación.

Descargar el compilador cruzado *arm-linux-gcc* y ejecutar los siguientes comandos:

```
cd /
bzcat arm-linux-gcc-3.3.2.tar.bz2 | tar -x
PATH=/usr/local/arm/3.3.2/bin:$PATH
```

Descargar la version de *QTDesigner* para la arquitectura *ARM* y ejecutar los siguientes comandos:

```
bzcat qt-embedded-free-3.1.2.tar.bz2 | tar –x
cd qt-embedded-free-3.1.2/
export QTDIR=`pwd`
./configure -embedded arm -qt-mouse-linuxtp –thread –no-exceptions -DQT_QWS_ADS –
DQT_QWS_IPAQ –DQT_NO_CUPS
gmake
```

Para compilar la aplicación:

```
PATH = /usr/local/arm/3.3.2/bin:$PATH
export QTDIR=/root/qt-embedded-free-3.1.2/
qmake –spec /root/qt-embedded-free-3.21.2/mkspecs/qws/linux-arm-g++ nombre.pro
make
```

Cuando se compile la aplicación, hay que indicar donde se encuentran las librerías y encabezados de la versión de OpenSSL para la arquitecutura ARM.

Ejecución.

Antes de ejecutar la aplicación en la tarjeta BitsyX se debe copiar el contenido de la carpeta *qt/lib* del directorio *qt-embedded-free-3.1.2* al directorio */* en la tarjeta y ejecutar los siguientes comandos:

```
export QTDIR=/qt
export LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$QTDIR/lib
export QWS_MOUSE_PROTO = linuxtp
```

Ejecutar los siguientes comandos en el directorio */qt/lib*:

```
ln –s libqte-mt.so.3.1.2 libqte-mt.so
```

```
ln –s libqte-mt.so.3.1.2 libqte-mt.so.3
ln –s libqte-mt.so.3.1.2 libqte-mt.so.3.1
```

Para utilizar la interfase con pantalla táctil se deben ejecutar los siguientes comandos en el directorio */dev*:

```
ln -s /dev/ts /dev/h3600_ts
ln –s/dev/ts_raw /dev/h3600_tsraw
```

Para poder ejecutar las aplicaciones que utilicen *OpenSSL* se debe descargar la versión para la arquitectura ARM y copiar el archivo *libcrypto.so.0.9.7* al directorio */usr/lib* en la tarjeta y ejecutar el siguiente comando:

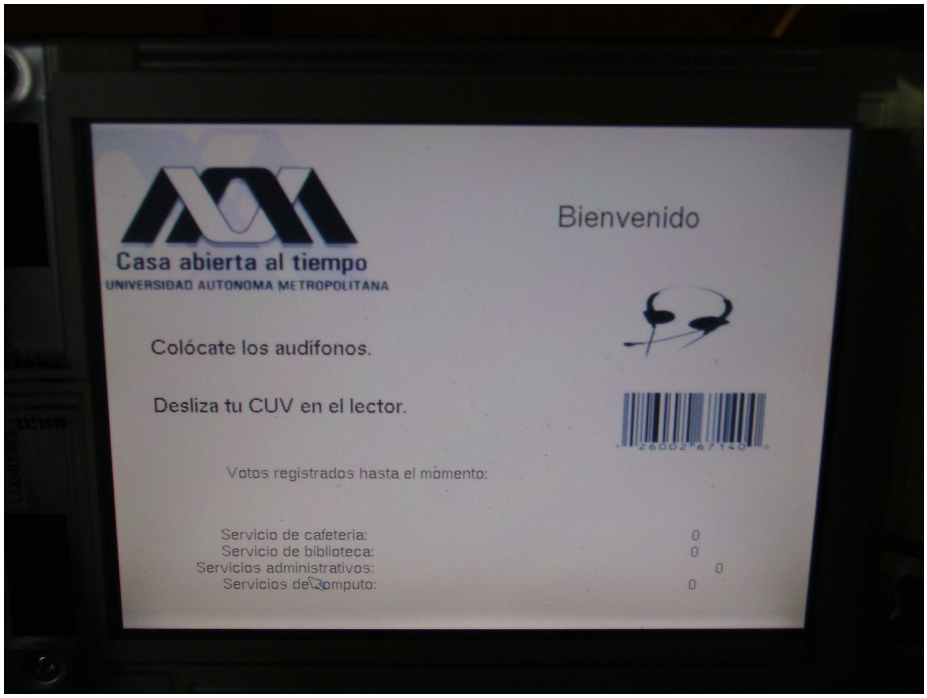
```
ln –s libcrypto.so.0.9.7 libcrypto.so
```

Finalmente para ejecutar la aplicación, se debe realizar de la siguiente manera:

```
./nombre –qws
```

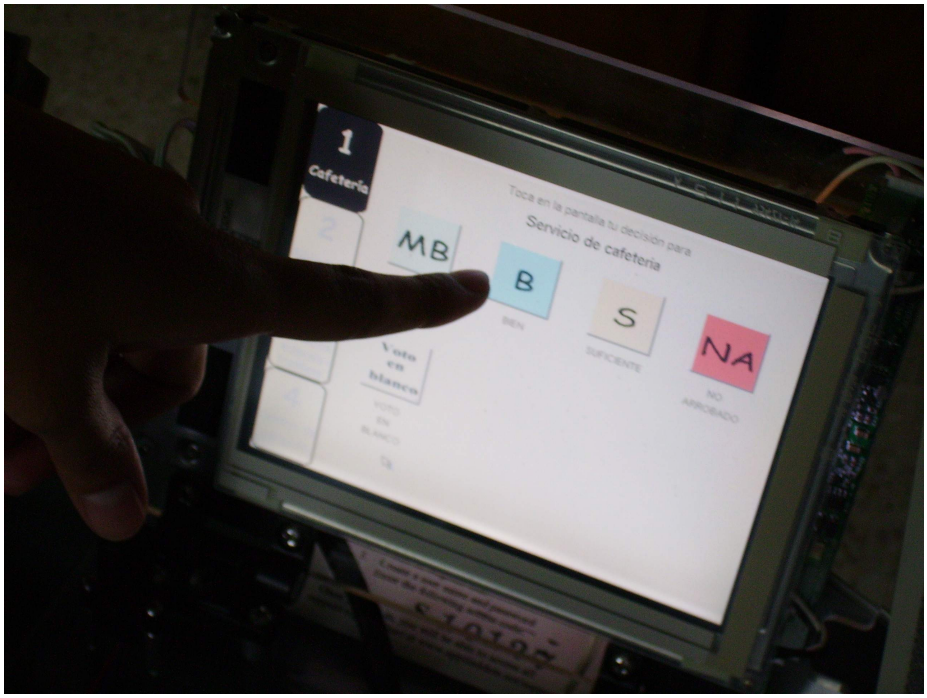
Capturas de la ejecución en el sistema BitsyX.

Captura 17. *Ejecución en la tarjeta BitsyX.*



Captura 18. *Pantalla de bienvenida.*

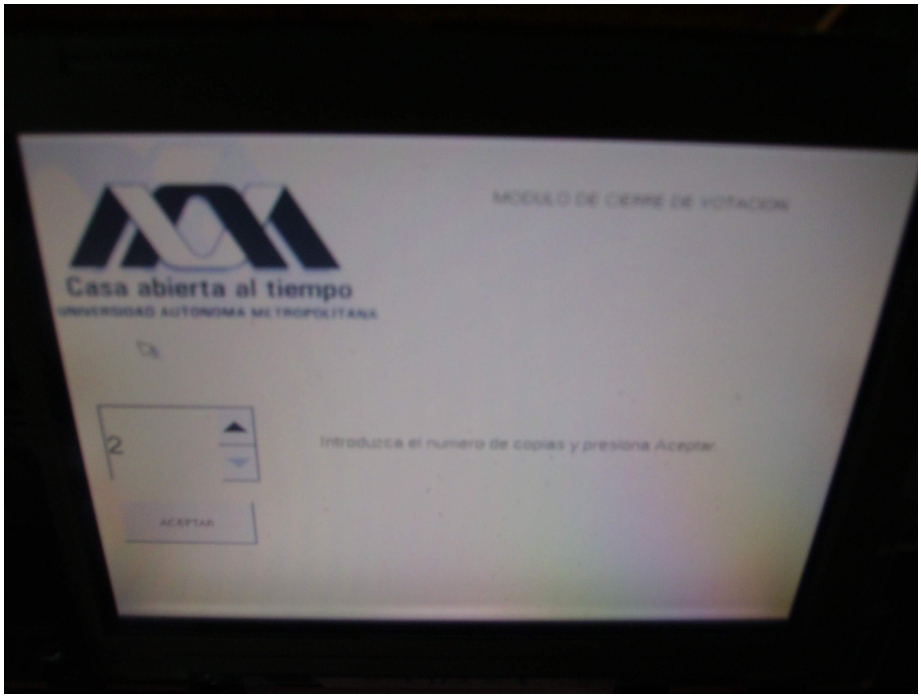
Captura 19. *Lector de código de barras y código del votante.*



Captura 20. *Boleta con interfase de pantalla táctil.*

Captura 21. *Pantalla de reporte de la contribución del voto.*

Captura 22. *Lector de código de barras y código del administrador.*



Captura 23*. Pantalla de impresión de actas finales.*

REFERENCIAS BIBLIOGRÁFICAS.

- [Arango & Sánchez, 2004]
  ARANGO, R. SANCHEZ, F. *Voto electrónico,* Todo Linux No 30, 2004.
- [Bidan, 1997]
  BIDAN, C. ISSARNY, V. *Security benefits from Software Architecture,* 1997.
- [Bolin, 2005]
  BOLIN, R. KATZ, E. *Electronic voting machines and the standards-setting process,* 2005.
- [Boneh, 1999]
  BONEH, D. *Twenty Years of Attacks Against the RSA Crypto-system,* Notices of the American Mathematics Society, 5(2), 1999.
- [Cranor & Cytron, 1997]
  CRANOR, L. CYTRON, R. *Secure Voting Systems,* Proceedings of the Hawai`s International Conference on System Sciences Hawaii U.S.A, 1997.
- [FEPADE, 2004]
  FEPADE, *La urna electrónica: Avances y Perspectivas,* Simposium sobre Urnas Electrónicas organizado por el IEDF realizado en México D.F, 2004.
- [Fischer, 2003]
  FISCHER, E. *Election reform and electronic voting systems (DREs): Analysis of Security Issues*, CRS Report for Congress, 2003.
- [García et al., 2004]
  GARCIA, L. MORALES, G. GONZALEZ, S. *Implementación del algoritmo RSA para su uso en el voto electrónico*, Simposium sobre Urnas Electrónicas organizado por el IEDF realizado en México D.F, 2004.
- [Graff, 2003]
  GRAFF, M. VAN W, K, *Secure Coding,* 2003.
- [IPL, 1997]
  IPL. *Software testing and software development lifecycles*, 1997.

- [Jones, 2004A]

  JONES, D. *Auditing Elections,* Communications of the ACM Vol 47. No 10, pp 46-50, 2004.

- [Jones, 2004B]

  JONES, D. *Parallel Testing: A menu of options*, 2004.

- [Kohno et al., 2004]

  KOHNO, T. STUBBLEFIELD, A. RUBIN, A. WALLACE, D. *Analysis of an Electronic Voting System*, IEEE Symposium on Security and Privacy. 2004.

- [Menezes, 1996]

  MENEZES, A. OORSCHOT, P. VANSTONE, S. *Handbook of Applied Cryptography.* 1996.

- [Mercuri, 2002]

  MERCURI, R. *A better ballot box*, 2002.

- [Prince, 2004]

  PRINCE, A. *Consideraciones, aportes y experiencias para el voto electrónico en Argentina*, Buenos Aires. 2004.

- [Probst, 2002]

  PROBST, S. ESSMAYR, W. WEIOOL, E. *Reusable Components for Developing Security-Aware Applications,* 2002.

- [Rivest, 2001]

  RIVEST, L. *Electronic Voting,* 2001.

- [Saltman, 2001]

  SALTMAN, R. *Auditability and Voter Confidence in Direct Recording (DRE) Voting Systems,* 2001.

- [Saltman, 2003]

  SALTMAN, R. *Auditability of non-ballot, poll-site voting systems,* 2003.

- [Selker, 2003]

  SELKER, T. GOLLER, J. *The SAVE System: Secure Architecture for Voting Electronically*, BT Technology Journal Vol 22 No 44, pp 89-95, 2003.

LIGAS DE INTERÉS.

OpenSSL.
http://www.openssl.org/

Flawfinder.
http://www.dwheeler.com/flawfinder/

RATS.
http://www.securesoftware.com/resources/download_rats.html

QTDesigner.
http://www.trolltech.com/download/opensource.html

QTEmbedded.
ftp://ftp.trolltech.com/qt/source/qt-embedded-free-3.1.2.tar.bz2

Compilador cruzado *arm-linux-gcc*
http://www.applieddata.net/developers/linux/files/tools/arm-linux-gcc-3.3.2.tar.bz2

Doxygen.
http://www.stack.nl/~dimitri/doxygen/index.html

Descarga de paquetes de *Linux Debian* para distintas arquitecturas.
http://www.debian.org/distrib/packages

Página de los fabricantes de la *BitsyX.*
http://www.applieddata.net/