Paakat: Revista de Tecnología y Sociedad
e-ISSN: 2007-3607
Universidad de Guadalajara
Sistema de Universidad Virtual
México
suv.paakat@redudg.udg.mx

Año 8, número 14, marzo-agosto 2018

Mecanismo de auditoría para detección de manipulación de votos en sistemas de votación electrónica

Audit Mechanism for Detecting Voting Manipulation in Electronic Voting Systems

Víctor Morales Rocha*
Universidad Autónoma de Ciudad Juárez, México

Óscar Ruiz Hernández**
Universidad Autónoma de Ciudad Juárez, México

Luis Felipe Fernández Martínez***
Universidad Autónoma de Ciudad Juárez, México

[Recibido 6/1/2018. Aceptado para su publicación 19/02/2018]
DOI: http://dx.doi.org/10.32870/Pk.a8n14.325

Resumen

Los sistemas de voto electrónico son propensos a fallas, errores y ataques intencionales. Un riesgo de estos sistemas es la adición o eliminación de votos directamente en el medio de almacenamiento, usualmente una base de datos. Existen diversos mecanismos diseñados para prevenir, detectar o mitigar las amenazas que pudieran atentar contra la integridad de los resultados de una elección que hace uso de medios electrónicos. Este trabajo presenta una revisión de los diversos mecanismos de auditoría para sistemas de voto electrónico. Además, se presenta una propuesta de generación y protección de logs en sistemas de voto electrónico remoto, con el fin de detectar manipulaciones en los votos almacenados.

Palabras clave

Voto electrónico; seguridad en voto electrónico; manipulación de votos; *logs*; auditoría.

Abstract

Electronic voting systems are prone to faults, errors and intentional attacks. One risk is the addition of fake votes (stuffing) that can be carried out by an external or internal entity. There are several audit mechanisms intended to prevent, detect and / or mitigate any threats that could undermine the election results. This paper presents a review and evaluation of the different mechanisms used to conduct an election audit for electronic voting systems. Also, an audit mechanism to detect ballot stuffing or any other manipulation on votes is presented.

Keywords

Electronic voting; electronic voting security; ballot manipulation; logs; audit.

Introducción

Un sistema de voto electrónico consiste en el uso de medios digitales en alguna de las fases de una elección, especialmente para la emisión del voto. El uso de los sistemas de voto electrónico, aunque lento va en aumento, debido principalmente a las ventajas que ofrece como la rapidez en el escrutinio, la accesibilidad para personas con discapacidades visuales y de otro tipo, la prevención de errores, entre otras. A pesar de esas ventajas, un sistema de voto electrónico también presenta desafíos considerables, especialmente en cuanto a seguridad se refiere, lo cual puede causar desconfianza en los votantes y demás participantes de una elección.

Se han propuesto mecanismos que permiten al votante comprobar que su voto se incluyó en el escrutinio, véase por ejemplo Chun-IFan y Wei-ZheSun (2008), así como Chaum *et al*. (2009). Un país pionero en el voto electrónico es Estonia, el cual utiliza un sistema de voto electrónico para realizar sus elecciones desde el año 2005. El mecanismo de votación de Estonia se compone de tres partes principales, como lo describen Mus, Sabir Kiraz, Cenk y Sertkaya (2016): 1) aplicaciones para el votante –una aplicación de votantes VoterApp y una aplicación de verificación VerifApp–; 2) un sistema central; 3) procesos de auditoría y recuento. El sistema de Estonia mediante la aplicación VerifApp permite verificar que el voto se haya emitido de acuerdo con la decisión del votante, dándole así mayor certeza al proceso electoral.

Bernhard *et al*. (2017) describen que aun cuando se incorporan mecanismos para corroborar el voto, es difícil asegurar que los resultados sean precisos. En las elecciones presidenciales de 2016 en Estados Unidos de América, donde aproximadamente tres cuartas partes de los estadounidenses votaron utilizando sistemas que generan registros en papel verificables por los votantes (VVPAT, Voter-Verified Paper Audit Trail), no se pudo proporcionar evidencia convincente de que los resultados que se informaron fueron precisos, o estuvieron libres de manipulaciones.

Por otro lado, si un atacante logra acceso a la base de datos donde se almacenan los votos, podría añadir votos ilegítimos, sin que los mecanismos de verificación individual puedan detectarlo. Ejemplo de ello ocurrió en las elecciones del estado de Florida (Estados Unidos de América) en el año 2012, cuando se agregaron más de 2 500 votos ilegítimos en poco tiempo.

De acuerdo con Alcaraz *et al*. (2012), el ataque se descubrió gracias a que el sistema informó de la detección de anomalías en patrones de tráfico, pues las solicitudes provenían de un rango de direcciones IP de Inglaterra, Irlanda e India. Si el ataque no hubiera sido detectado, el número de votos falsos habrían sido suficientes para afectar el resultado de las elecciones. La detección de este tipo de ataques queda fuera del alcance del votante, es decir, aun cuando cada votante puede verificar que su voto se ha registrado correctamente no detectaría la adición de votos ilegítimos.

Es entonces necesario implementar mecanismos de auditoría que faciliten la detección de prácticas fraudulentas en los sistemas de votos electrónicos, más allá de la verificación individual. En las siguientes secciones se describen y analizan las diversas técnicas y mecanismos de auditoría que se utilizan al concluir un proceso de elección, las cuales están diseñadas para detectar prácticas que podrían alterar el resultado; se hace especial énfasis en los mecanismos de protección de *logs*. Posteriormente, se describe una propuesta de auditoría para la detección de manipulación de votos en sistemas de voto electrónico.

Mecanismos de auditoría post-elección

Una auditoría posterior a la elección pretende verificar o validar el resultado de la elección, así como detectar si hubo cualquier tipo de error o manipulación que pudiera haber alterado dicho resultado. En esta sección se describen y analizan diversos mecanismos de auditoría.

Recuento total de votos

No es de extrañar que, en el voto tradicional donde se utilizan boletas de papel, se presenten errores en el conteo, pues los votos son contados por humanos. Debido a eso, si se lleva un recuento total existe una probabilidad alta de que se encuentren diferencias entre el conteo original y el recuento. En sistemas de voto electrónico el escrutinio de votos se realiza de manera automática; un recuento total de votos arrojará el mismo resultado que el original, asumiendo que el sistema de conteo automatizado no presenta errores en su programación. Por tanto, un recuento total de votos en el entorno del voto en papel dista mucho de ser fiable, y un recuento total en un sistema de voto electrónico no ofrecerá alguna pista de posibles manipulaciones previas al conteo.

Recuento total con medios independientes

Si nos enfocamos en sistemas de voto electrónico presencial, una auditoría se puede realizar a través de un método de verificación independiente,[1] como los descritos en *Voluntary Voting Systems Guidelines* (VVSG, 2005), los cuales pueden ser utilizados para fines de verificación individual,[2] así como para llevar a cabo una auditoría. Estos sistemas de verificación independiente tienen la finalidad de almacenar en tiempo real los votos en un medio alternativo a fin de contar con un respaldo, por lo cual es factible hacer un recuento total del registro realizado.

Un recuento a través de un medio independiente presenta, sin embargo, una desventaja considerable. Si el recuento da diferentes resultados a los originales, surge el problema de decidir cuál de los dos resultados es más fiable, cualquiera de los dos pudo haber sido manipulado. Por tanto, este tipo de sistemas de auditoría presenta el desafío de escoger el registro (generado por el sistema de votación o el registro independiente), el cual será considerado válido en caso de que existan diferencias. Más allá de lo tecnológico es, quizás, un tema de legislación electoral que anticipe tales situaciones.

Recuento de una muestra de votos

Los recuentos parciales de votos son un método de auditoría frecuentemente utilizado en los sistemas electorales de voto en papel. Se toman como unidades sujetas a auditoría los recintos de votación, o bien, una unidad electoral más grande como puede ser un distrito. Por su parte, en los sistemas de voto electrónico presencial la unidad mínima sujeta a auditoría suele ser una máquina de votación, o en algunos casos donde el sistema de votación imprime un respaldo en papel, la unidad mínima de auditoría puede ser el recinto donde se localizan varias máquinas de votación.

En la mayoría de los países o estados, la decisión de realizar un recuento total o parcial dependerá de una legislación o reglamento de procedimientos electorales, así como de la situación específica que requiera de una auditoría, por ejemplo, la razón de una auditoría puede ser que la diferencia entre los votos a favor de un candidato y de otro sea mínima, o bien debido a un proceso de impugnación de parte de uno o más de los partidos o candidatos participantes. En algunos estados de los Estados Unidos de América se debe realizar un recuento parcial como procedimiento estándar después de una elección. Por ejemplo, en Colorado se debe auditar 5% de las máquinas utilizadas; en Maryland se debe auditar al menos 10% de los recintos de votación. Para más ejemplos de parámetros y procedimientos de auditorías se puede consultar Verified Voting (2017).

Las ventajas de un recuento parcial frente a un recuento total son evidentes. Por un lado, el costo de recursos y tiempo para realizar la auditoría es considerablemente menor, además de los inconvenientes presentados en los recuentos totales, como ya se explicó previamente. A través de un recuento parcial donde se seleccionan apropiadamente los parámetros para determinar la muestra a auditar, se puede identificar con una precisión alta si ha existido alguna manipulación.

Para determinar una muestra fiable se hace uso de modelos estadísticos y probabilísticos que determinan el porcentaje o cantidad de votos, máquinas de votación o recintos de votación o de cualquier otra unidad que se determine, la cual debería ser auditada a fin de detectar, con un nivel de confianza alto, si han existido manipulaciones que impactan en los resultados de la elección. Entre estos trabajos podemos encontrar los de Stanislevic (2006), Rivest (2006), Stark (2010), Bretschneider *et al*. (2012), Stark (2015) y Rivest (2017). Algunas variables generalmente se toman en cuenta, además del tamaño de la elección, son la diferencia de votos entre los candidatos, el tamaño del censo en los diferentes recintos o máquinas de votación, entre otras.

Se debe tener en cuenta que los recuentos de votos, ya sea parciales o totales, solamente detectarán manipulaciones o errores cometidos en el conteo inicial. Sin embargo, si se han alterado o eliminado los votos en sí (ya sea en papel o registros digitales) es difícil detectar dichas manipulaciones con este tipo de auditorías. Por otro lado,

en sistemas de voto electrónico remoto, es decir, aquellos en los cuales se utiliza Internet como medio de transmisión de los votos no es factible realizar recuentos parciales, pues el registro de votos se realiza en una base de datos centralizada, o en algunos casos, en unas cuantas bases de datos distribuidas en diferentes puntos físicos; no hay una gran cantidad de unidades sujetas a auditar. Por otro lado, como ya se comentó previamente un recuento a través del mismo medio automatizado, como es el caso de un sistema de votación electrónico remoto, ofrecerá el mismo resultado del conteo original.

Sistemas de protección y verificación de logs

Un mecanismo para realizar auditorías en sistemas de voto electrónico, tanto presencial como remoto, es la auditoría a nivel de *logs*. Un *log* es un registro de un evento que sucede en un entorno digital a manera de bitácora, asociando el evento a una fecha y hora, usuario o proceso que lo realizó, entre otras características que interese registrar. Los sistemas operativos generalmente llevan un registro de *logs*, es decir, de todos los eventos que suceden en un equipo de cómputo. Además, los sistemas de información o aplicaciones que tienen funciones críticas tales como accesos a bases de datos, actualización de información, etcétera, también suelen llevar un registro de los eventos específicos de ese sistema, precisamente con fines de auditoría.

En los sistemas de voto electrónico es sumamente importante tener un registro de los eventos, a fin de contar con las evidencias de las acciones realizadas especialmente para los casos donde se sospeche que ha habido manipulaciones. En una sesión de voto, por ejemplo, se puede generar el registro de la autenticación del votante, el evento de la selección del o de los candidatos, el cierre de la sesión, etcétera. Asimismo, es importante tener el registro de eventos a nivel base de datos, es decir, cada una de las transacciones que se realizan directamente en la base de datos sin tener que utilizar el sistema de información.

A pesar de las ventajas que ofrece la generación de *logs* no cubre por completo la necesidad de tener un mecanismo certero para auditoría, pues un atacante que ha manipulado una elección, suponiendo que sea un usuario con privilegios de acceso, también podría manipular los *logs* a fin de eliminar cualquier prueba de la manipulación. Se han propuesto técnicas que permiten detectar, en cierto grado, las manipulaciones realizadas en los *logs* de un sistema. A continuación, se describen los trabajos más relevantes para la protección de *logs*.

En el trabajo descrito por Bellare y Yee (1997) se presenta un mecanismo de protección de *logs* contra manipulaciones. El concepto principal de este trabajo es que a pesar de que los *logs* sean accedidos por un usuario o atacante externo, este no pueda realizar modificaciones sin la posibilidad de dejar rastro. Por otro lado, si los contenidos fueran eliminados, el mecanismo permite detectarlo. Este protege los *logs* a través de funciones MAC o códigos de autenticación de mensajes, los cuales utilizan claves privadas.

La seguridad de este mecanismo se fundamenta en el hecho que es computacionalmente inviable que un atacante sin conocer la clave privada altere el mensaje, y que aun así al comparar el resultado de aplicar la función MAC resulte el mismo valor. A fin de evitar que un atacante con acceso a la clave privada en un momento dado manipule los *logs* generados hasta ese momento, la clave privada puede cambiar a través

del tiempo. La clave privada *ki* en un tiempo *ti* se obtiene aplicando una función *hash* a la clave *ki-1* que pertenece a un tiempo previo *ti-1*, y una vez iniciado el tiempo *ti* esa clave *ki-1* es eliminada. De manera que si el atacante obtiene una clave privada *ki* no podrá conocer o deducir una clave privada *kj* para $j < i$. De esta manera, los *logs* generados antes de comprometer la clave privada no pueden ser modificados sin detección.

El principal inconveniente de este mecanismo es la administración de las claves privadas, lo cual se vuelve más complejo a medida que aumenta la cantidad de claves. Para verificar la integridad de los *logs* se deben usar todas las claves privadas que fueron generadas para la protección. Además, solo se contemplan los ataques de modificación de *logs*, mas no la eliminación de estos.

Schneier y Kelsey (1998) proponen otro mecanismo de protección que tiene como objetivo que los *logs* se vayan respaldando después de cierto tiempo, o cuando se alcance cierto número de registros. En este mecanismo se consideran tres componentes: una máquina insegura, una máquina segura y un verificador. En la máquina insegura es donde se generan los *logs*, y el respaldo se realiza en la máquina segura. Se asume una conexión entre las dos máquinas a través de una red de datos. Se hace uso de una combinación de funciones *hash*, firmas digitales y criptografía de clave pública y privada. Al igual que el mecanismo descrito previamente, este también se enfoca en la protección de los *logs*, los cuales se generan antes del compromiso.

Otras propuestas de protección de *logs*, principalmente orientadas a sistemas de voto electrónico, son las descritas por Riera y Puiggali (2004) y por Sandler, Derr, Crosby y Wallach (2008). Riera y Puiggali describen un protocolo de protección de *logs* para sistemas de voto electrónico remoto. La idea consiste en ir calculando en tiempo real un valor *hash* de cada *log* generado. Además, cada cierto número de *logs* acumulados, o bien, cada período específico se aplica una firma digital. Se realiza entonces una cadena de *logs* de la siguiente manera:

$$\ldots$$
$$L_i = H\,(l_i \,|\, L_{i-1})$$
$$L_{i+1} = H\,(l_{i+1} \,|\, L_i)$$
$$L_{i+2} = H\,(l_{i+2} \,|\, L_{i+1})$$
$$L_{i+3} = H\,(l_{i+3} \,|\, L_{i+2})$$
$$\ldots$$

Donde *l* es un *log* y *L* es el resultado del valor *hash* de la concatenación de *l* con el *L* previo. Cada vez que se forma un bloque *bi* se lleva a cabo la firma digital de dicho bloque:

$$Sig_i = [H\,(b_i \,|\, sig_{i-1})]S_k$$

Donde S_k es la clave privada del equipo de cómputo o servidor que firma los *logs*. A fin verificar la integridad de dichos *logs* se utiliza la clave pública correspondiente. La idea

de componer una cadena de integridad entre los *logs* es que en caso de que un atacante modifique los *logs* esto será detectado al realizar la verificación de las firmas. Asimismo, si un *log* es eliminado esto también sería detectado al realizar la verificación del bloque al cual pertenecía.

Es importante resaltar que cuando la cantidad de eventos es muy grande, resulta sumamente complejo llevar a cabo un análisis de los registros de *logs*, lo cual deriva en la dificultad de detectar manipulaciones. Para realizar un análisis de *logs*, especialmente cuando hablamos de grandes cantidades de registros, se requiere de herramientas que automaticen dicho análisis.

Sandler, Derr, Crosby y Wallach (2008) describen una herramienta llamada *Querifier*, la cual realiza el análisis de *logs* en tiempo real, esto para disminuir la complejidad de la tarea que representa el análisis de grandes cantidades de registros. Esta herramienta hace uso de reglas definidas con lógica de predicados, de manera que se definen todos los eventos posibles en el sistema y el orden más lógico en el cual deberían suceder dichos eventos. Por ejemplo, en un sistema de voto electrónico una regla podría ser que el evento "voto registrado" debería estar precedido por el evento de "votante autenticado", de forma que si no se cumple la regla se detectaría que ha habido alguna corrupción en los registros de eventos. También se contempla detectar si se ha roto una cadena de integridad realizada a través de funciones *hash*, de manera que se puede saber cuándo se ha realizado alguna manipulación en el *log*.

Al haber realizado un análisis de los mecanismos previamente descritos para realizar auditorías en los sistemas de voto electrónico remoto, se pudo notar que estos esquemas basan la detección de las manipulaciones, principalmente, en el análisis de los *logs* generados. De acuerdo con los esquemas descritos, la dificultad radica no solo en la protección de los *logs*, sino en la verificación de la integridad, lo cual es una tarea que deben realizar los auditores y que sin la ayuda de herramientas de análisis de *logs* dicha tarea es prácticamente imposible de desarrollar.

En la propuesta descrita en Morales (2009) se describe un mecanismo específico para detectar la adición de votos (ataque conocido en la literatura como *stuffing*) por parte de un atacante interno, es decir, con privilegios de acceso a los elementos de la elección o al menos a la base de datos donde se almacenan los votos. Dicha propuesta describe una técnica de protección de votos almacenados, la cual permite realizar auditorías de una manera más simple en comparación a un análisis de *logs*.

La técnica permite generar bloques o lotes de los votos para respaldo en tiempo real, es decir, a medida que estos son recibidos en un servidor se define el tamaño que debe tener cada lote, la cantidad de votos que lo deben componer, de manera que cuando se completa uno los votos se concatenan y se firman digitalmente por una autoridad o por un conjunto de autoridades que poseen una clave Sk, de la siguiente manera:

$$L = \{V1 | V2 | V3 | \ldots | Vn\}Sk$$

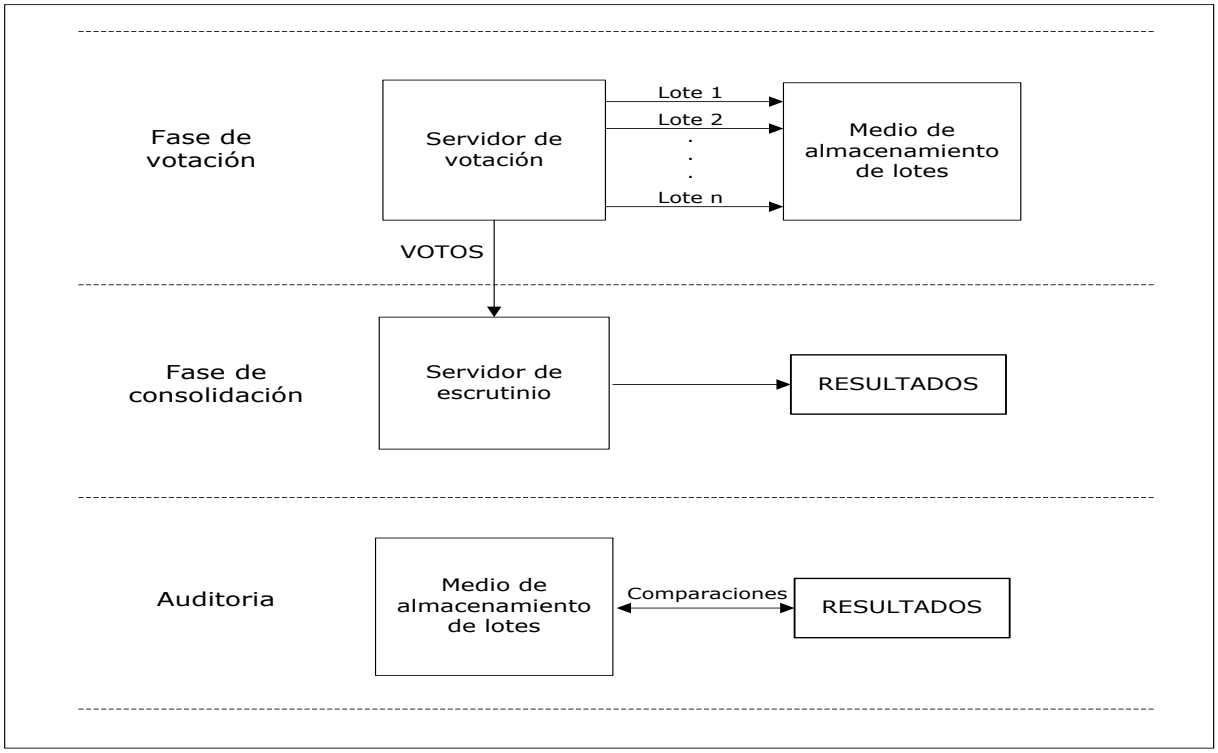
Cada uno de los lotes de votos se almacenan en un servidor de respaldo o en algún medio de almacenamiento extraíble, a fin de mantenerlos en un lugar seguro. A través de la firma digital se pueden detectar manipulaciones en cualquiera de los lotes. Además, a fin de contrarrestar un posible ataque en el cual se pudieran añadir lotes ilegítimos e incluso

firmarlos si se diera el caso de que el atacante posea la clave privada, se forma una cadena de integridad de los lotes a medida que estos se van generando. Para realizar la cadena de integridad se calcula un valor de *hash* de la concatenación del lote previo, ya firmado con el lote actual también firmado de esta manera:

$$L'1 = H\,[L1]$$
$$L'2 = H\,[L'1 \,|\, L2]$$
$$\ldots$$
$$L'n = H\,[L'n\text{-}1 \,|\, Ln]$$

Para fines de auditoría se cuenta con un identificador único de voto, de manera que se puedan relacionar los votos contenidos en los lotes con los votos almacenados en la base de datos. La generación del identificador único de voto se realiza durante una sesión de voto autenticada. De tal manera que si se agregan votos ilegítimos en la base de datos, lo cual se haría sin una sesión de votante válida, esos votos no tendrán un identificador válido. Para realizar una auditoría se lleva a cabo una serie de validaciones comparando los votos incluidos en el escrutinio con los votos contenidos en los lotes de respaldo. La figura 1 muestra este esquema de manera general:

Figura 1. Esquema de auditoría



Fuente: Morales, 2009.

En las validaciones realizadas en la auditoría se hace lo siguiente:

- Verificación de la integridad de los lotes. Se revisa la cadena de integridad de los lotes y la firma digital de cada uno de ellos.
- Verificación de la cantidad de votos. Se efectúa la comparación de la cantidad de votos incluidos en el escrutinio con el número de votos contenidos en los lotes de respaldo. Si la cantidad de votos que se incluyeron en el escrutinio es mayor que el de votos contenidos en los lotes se puede saber que se han añadido votos ilegítimos. Aun así se debe realizar el siguiente paso para detectar cuáles son los votos ilegítimos.
- Verificación del identificador de voto. Se valida que cada voto incluido en el escrutinio se encuentra también en alguno de los lotes, lo cual se logra verificando los identificadores únicos de los votos. Los votos del escrutinio, cuyo identificador no coincida con alguno de los votos registrados en los lotes de respaldo, se pueden clasificar como votos ilegítimos.

Después de realizar estas validaciones en el proceso de auditoría se puede detectar si se han añadido votos o si se han manipulado los votos almacenados en la base de datos original. Este mecanismo también permite detectar cuáles han sido los votos manipulados.

Propuesta de auditoría para detección de manipulación de votos: Medusa

Con base en lo descrito en Morales (2009), analizado en la sección previa, se propone un mecanismo que incorpora algunas mejoras para la detección de la adición de votos ilegítimos y la eliminación de votos legítimos. Este mecanismo se denomina Medusa y, a grandes rasgos, las mejoras son: 1) el uso de *hash* para formar la cadena de integridad en lugar de la firma digital; y 2) la identificación del último lote para corroborar su existencia e integridad. En los siguientes párrafos se describe a detalle su funcionamiento.

Al igual que la propuesta anterior se generan lotes de respaldo de los votos a medida que estos son recibidos por el servidor de votación. Un lote de votos se conforma de cierta cantidad de votos recibidos que se van concatenando, se les aplica una función *hash* y son firmados por una autoridad o conjunto de autoridades con la clave Sk, como se muestra a continuación:

$$L = \{V1|V2|V3|\ldots|Vn\} \mid (H\{V1|V2|V3|\ldots|Vn\})Sk$$

Por ejemplo, si se determina que los lotes deben contener 100 votos estarían conformados de la siguiente manera:

$$L1: \{V1, \ldots, V100\} \mid (H\{V1, \ldots, V100\})Sk$$
$$L2: \{V101, \ldots, V200\} \mid (H\{V101, \ldots, V200\})Sk$$
$$\ldots$$

$$Ln: \{V100n-99, \dots, V100n\} \mid (H\{V100n-99, \dots, V100n\})Sk$$

Los lotes de votos firmados se almacenan en un servidor o medio diferente al servidor de votación. Con la implementación de la firma digital se asegura que, en caso de presentarse alguna manipulación en los lotes, esta pueda ser detectada. Aunque existe la posibilidad de que si el atacante se apodera de la clave privada pueda agregar y eliminar lotes sin ser detectados. Si los ataques son perpetrados al realizar una auditoría los votos contabilizados originalmente no concordarían con los votos respaldados en lotes.

Para prevenir estos ataques se forma una cadena de integridad entre los votos y lotes a medida que se generan. Esta cadena de integridad se realiza calculando un *hash* de la concatenación del lote anterior firmado con el primer voto del lote actual, y entre los votos se realiza calculando un *hash* a la concatenación del voto previo con el voto actual:

$$\dots$$
$$Vi$$
$$V'i = H(Vi \mid Li\text{-}1)$$
$$Vi+1$$
$$V'i+1 = H(Vi+1 \mid Vi)$$
$$Vi+2$$
$$V'i+2 = H(Vi+2 \mid Vi+1)$$
$$Vi+3$$
$$V'i+3 = H(Vi+3 \mid Vi+2)$$
$$\dots$$

Donde:

- Li-1 es el lote que contiene votos, *hashes* y firma correspondiente del lote anterior.
- V es un *log* individual.
- V' es el resultado de calcular un valor *hash* a la concatenación del voto actual con el voto o lote previo, es decir Vi con Vi-1.

Como ya se mencionó, una mejora incluida en Medusa es que para formar la cadena de integridad no se hace uso de la concatenación de la firma. En cambio, se hace uso del cálculo de un *hash* de la concatenación del primer voto y el lote anterior. Al implementar esto se obtienen las siguientes ventajas:

- Se requiere de menor poder computacional para la creación y comprobación de la cadena de integridad, ya que se le aplica una función *hash* a la concatenación en lugar de aplicar una firma digital que requiere más procesamiento.
- No es necesario calcular la firma digital de cada lote para comprobar su integridad, ya que solo será calculada en aquellos lotes donde se rompe la cadena de integridad. Por ejemplo, si se tiene L1, L2, L3, L4 y L5, si la cadena de integridad se rompe en

L3 solo es necesario comprobar la integridad mediante firma digital de L2 y L3 para tratar de indagar si se realizó alguna modificación o se eliminó algún lote.

También este mecanismo propuesto considera la posibilidad de detectar cuando el último lote o lotes son eliminados. Por ejemplo, si se tienen cinco lotes L1, L2, L3, L4 y L5 y se elimina L5, la cadena de integridad no se rompe y no podría ser detectada la eliminación. Otro caso que se podría presentar es si se desea eliminar L3 sin dejar rastro simplemente se elimina L3, L4 y L5, y esto no podría ser detectado.

Entonces para tratar de mitigar este riesgo se propone identificar el último lote, ofreciendo así la posibilidad de detectar si fue eliminado o no. Para la identificación del último lote se propone realizar una firma extra, es decir, además de aquella que se realiza para asegurar la integridad del lote se hace una firma adicional para comprobar que es el último. Es decir, se firma la concatenación del penúltimo lote con el último de la siguiente manera:

$$\text{Sig n} = \{H(Ln\text{-}1 \mid Ln)\}Sk$$

Donde Sk es la clave privada de una autoridad o de un conjunto de autoridades de la elección a cargo de firmar digitalmente los lotes. La clave pública correspondiente será usada para verificar la integridad y asegurarse que exista el último lote donde es almacenado. También se debe de asegurar que solo el último lote esté marcado, así que cuando este se identifica es necesario quitar la marca al penúltimo de los lotes para evitar confusiones.

Al igual que el mecanismo propuesto en Morales (2009), al realizar una auditoría es necesario que los votos contengan un identificador único a fin de establecer una relación entre los votos contenidos en los lotes y los votos almacenados en el servidor de votación, y de esa manera poder compararlos en una auditoría. El identificador del voto es generado en la fase de votación y puede ser compuesto de un conjunto de caracteres alfanuméricos.

El identificador único también puede servir para integrar los lotes de respaldo de acuerdo a la unidad electoral a la cual pertenecen, por ejemplo, a un municipio, distrito electoral, estado, etcétera. Por ejemplo, si a un distrito electoral se le asigna el identificador "001", los votos emitidos por los votantes de dicho distrito podrían ser identificados con "001" además de otro conjunto de caracteres únicos. Esto serviría para detectar manipulaciones dirigidas a un área geográfica específica, a fin de favorecer a un candidato particular.

Al igual que el mecanismo anterior solo hasta que el votante ha sido autenticado puede llevarse a cabo la generación de un identificador de voto durante una sesión de votación. Al implementar lo anterior se puede contar con la certeza de que si se agregaran votos no válidos en la base de datos, estos no contarán con un identificador válido; asimismo, si hay votos que son eliminados se podría identificar cuáles fueron. En ambos casos los votos podrían identificarse al realizar un cruce entre los lotes de respaldo y la base de datos.

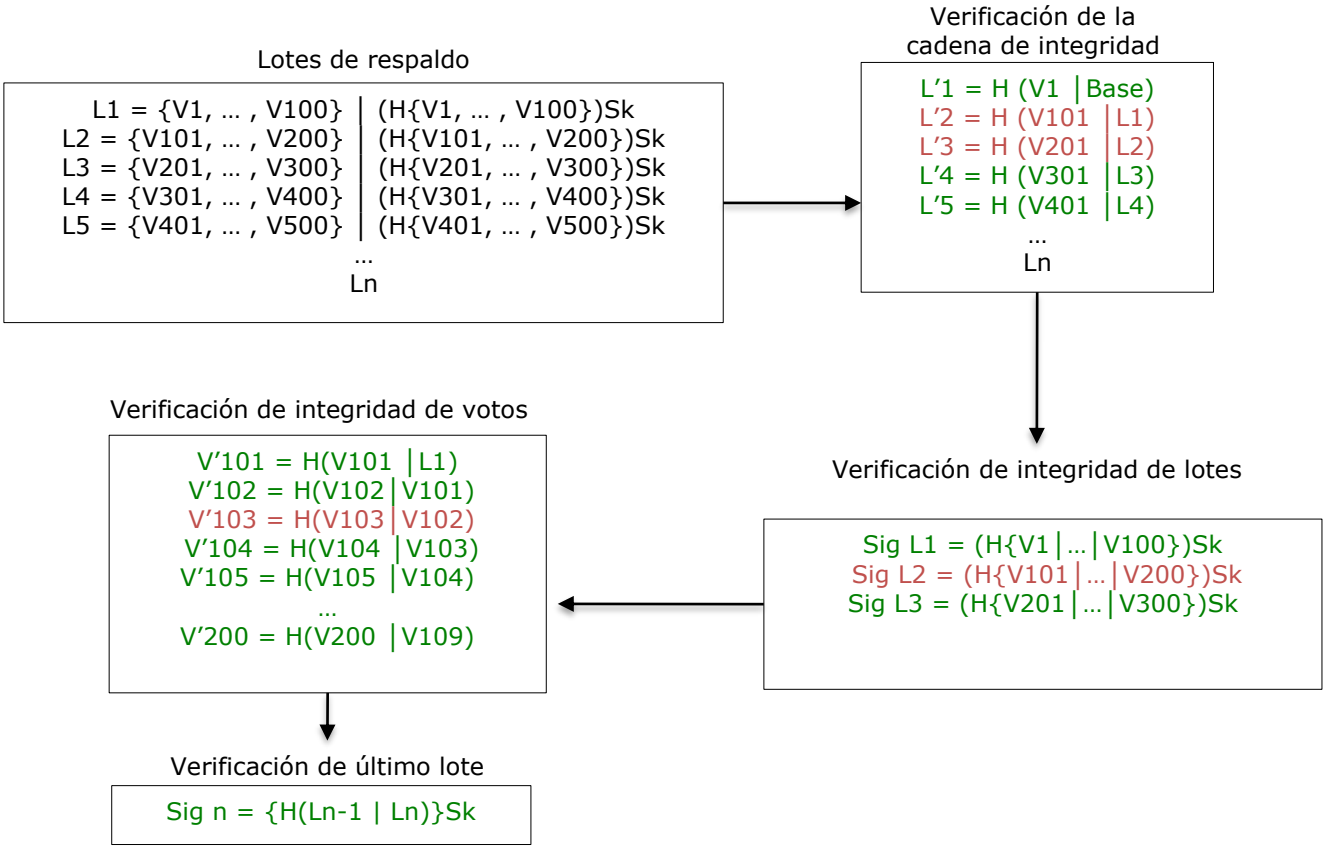
Se debe tener en cuenta que un atacante puede añadir votos ilegítimos en tres fases del proceso de elección: antes de iniciar el proceso de votos, durante la fase de votación o,

una vez concluida esta. En cambio, la eliminación de votos legítimos se puede presentar solo en dos períodos: en la fase de votación y una vez que ha terminado esa fase. La adición de votos antes de iniciar el proceso de votación puede ser detectada de forma relativamente fácil, por medio de verificaciones. Por ejemplo, se verifican que la base de datos donde serán almacenados los votos se encuentra en blanco antes de iniciar la votación. La adición y eliminación de votos en la propia fase de votación e incluso cuando esta ha terminado, se puede detectar con el mecanismo de auditoría propuesto como se puede observar en las siguientes líneas.

Si una vez que se concluye el conteo de votos es necesario llevar a cabo una auditoría de los resultados de la votación se pueden realizar algunas validaciones al comparar los votos incluidos en el conteo original con los votos almacenados en los lotes de respaldo, los cuales se encuentran protegidos contra manipulaciones. La auditoría hace uso del mecanismo propuesto, y se lleva a cabo realizando los siguientes pasos:

1) Al igual que el mecanismo anterior se verifica la integridad de los lotes. En la figura 2 se muestra un ejemplo de las validaciones realizadas para detectar si un voto ha sido modificado. Se verifica la cadena de integridad de los lotes y la firma digital donde la cadena se rompa para poder detectar:
    - Si el primer lote o un conjunto de los primeros lotes fueron eliminados: si la cadena de integridad se rompe en el primer lote y se comprueba mediante la firma digital que el primer lote no ha sufrido modificaciones se puede determinar que ha sido eliminado un lote o un conjunto de lotes del inicio.
    - Si el último lote o un conjunto de los lotes finales fueron eliminados: debido a que se identifica el último lote se puede calcular la firma digital para corroborar que existe y no ha sido modificado.
    - Si un lote o conjunto de lotes intermediarios fueron eliminados: en el punto de quiebre de la cadena de integridad si se comprueba que el lote anterior y actual no sufrieron modificaciones mediante la firma digital se puede determinar que algún lote o conjunto de lotes fueron eliminados.
    - Si un voto fue modificado o eliminado: cuando la cadena de integridad se rompe y se comprueba que un lote fue modificado mediante la firma digital se puede encontrar que un voto fue modificado o eliminado calculando el *hash* de la concatenación del voto actual con el voto anterior.
    - Se puede precisar en cuáles lotes se rompe la cadena de integridad: al calcularse el *hash* de la concatenación del primer voto con el lote anterior para comprobar si se conserva o se ha roto la cadena de integridad y así determinar en qué punto o puntos se rompe.
    - Se puede precisar en cuáles votos se rompe la cadena de integridad de cada lote: se calcula el *hash* de la concatenación del voto actual con el voto anterior para comprobar si se conserva o se ha roto la cadena de integridad y así determinar en qué punto o puntos se rompe.

Figura 2. Ejemplo de verificación de integridad



Fuente: elaboración propia.

2) Se compara el número de votos obtenidos del escrutinio con el número de votos en los lotes de respaldo. Si el número de votos del escrutinio es diferente al de los votos contenidos en los lotes se puede deducir que se han añadido votos ilegítimos, o se han eliminado votos legítimos. Se debe de realizar el siguiente paso pasa saber cuáles votos fueron añadidos o eliminados.

3) Gracias a que todos los votos cuentan con un identificador se puede verificar que cada voto incluido en el escrutinio se encuentra también en alguno de los lotes. Los votos que no se encuentren en los lotes de respaldo se clasifican como votos ilegítimos. Los votos del escrutinio, cuyo identificador no se encuentre en alguno de los lotes de respaldo se catalogan como votos eliminados.

A través de estas validaciones se puede obtener evidencia de si han sido añadidos votos ilegítimos o eliminado votos legítimos en la base de datos original, e incluso pueden ser identificados dichos votos falsos. De ser necesario puede ser llevado a cabo un nuevo escrutinio de los votos.

Al llevar a cabo una auditoría pueden ser detectadas algunas evidencias que sugieren que ha habido algunas manipulaciones, y que posiblemente requieran un análisis más profundo que el proporcionado por el mecanismo propuesto. Si ese fuera el caso,

entonces se puede implementar alguno de los mecanismos que permiten realizar un análisis de *logs*, tal como ha sido explicado anteriormente.

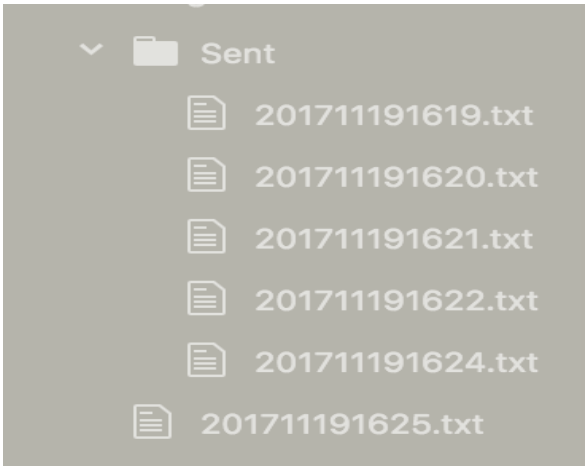
Prototipo de Medusa

El mecanismo propuesto está enfocado a la detección de adición y/o eliminación de votos ilegítimos, sin embargo, puede funcionar igualmente para la protección y detección de manipulaciones en los *logs* que generan algunos sistemas de información que manejan información sensible. En el Laboratorio Nacional de Tecnologías de Información de la Universidad Autónoma de Ciudad Juárez se desarrolló un prototipo, basado en el mecanismo Medusa, para la generación y protección de *logs*, así como la verificación de la integridad de estos.

El prototipo formó parte del proyecto ERRS (*Elections Results Reliable Sampling*), orientado a la verificación de resultados electorales. Este prototipo fue desarrollado en *JavaScript* y se usó *Node.js* como entorno de ejecución. El prototipo se realizó como se explica a continuación:

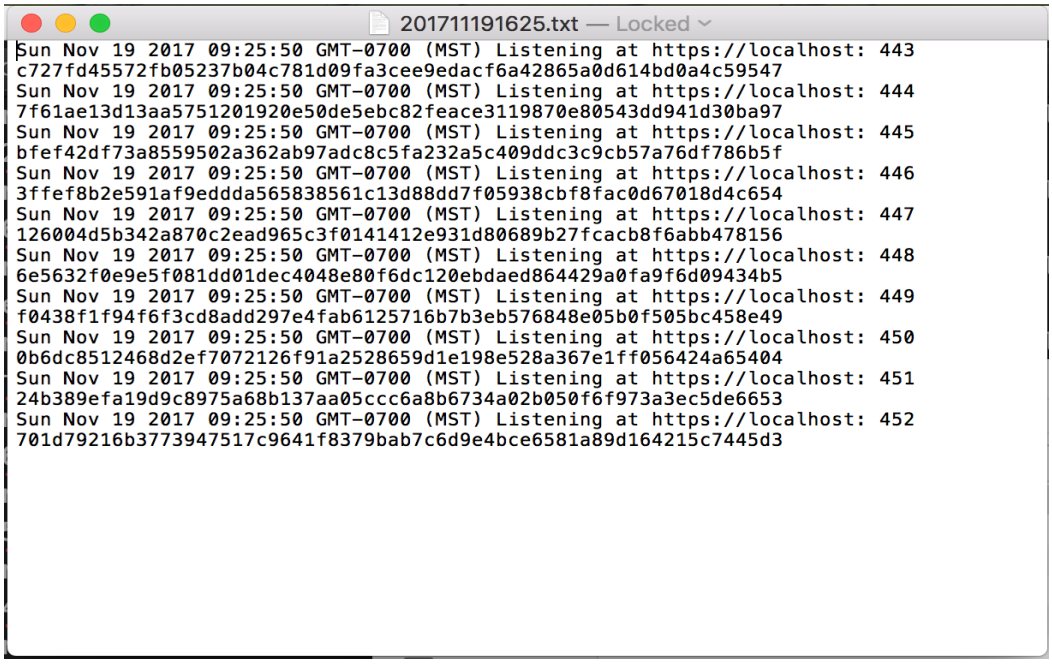
1) Al generar un nuevo *log* también es generado un archivo de texto por cada minuto de funcionamiento del sistema, por ejemplo "201708170212.txt", donde serán almacenados los *logs* –ver figura 3–. Al generar un *log* se calcula en tiempo real un valor *hash* de la concatenación del l*og* generado con el *log* anterior, esto con el objetivo de generar una cadena de integridad entre los *logs*, los archivos de *logs* tienen la estructura que se muestra en la figura 4:

Figura 3. Archivos *logs*



Fuente: elaboración propia.

Figura 4. Estructura de archivos



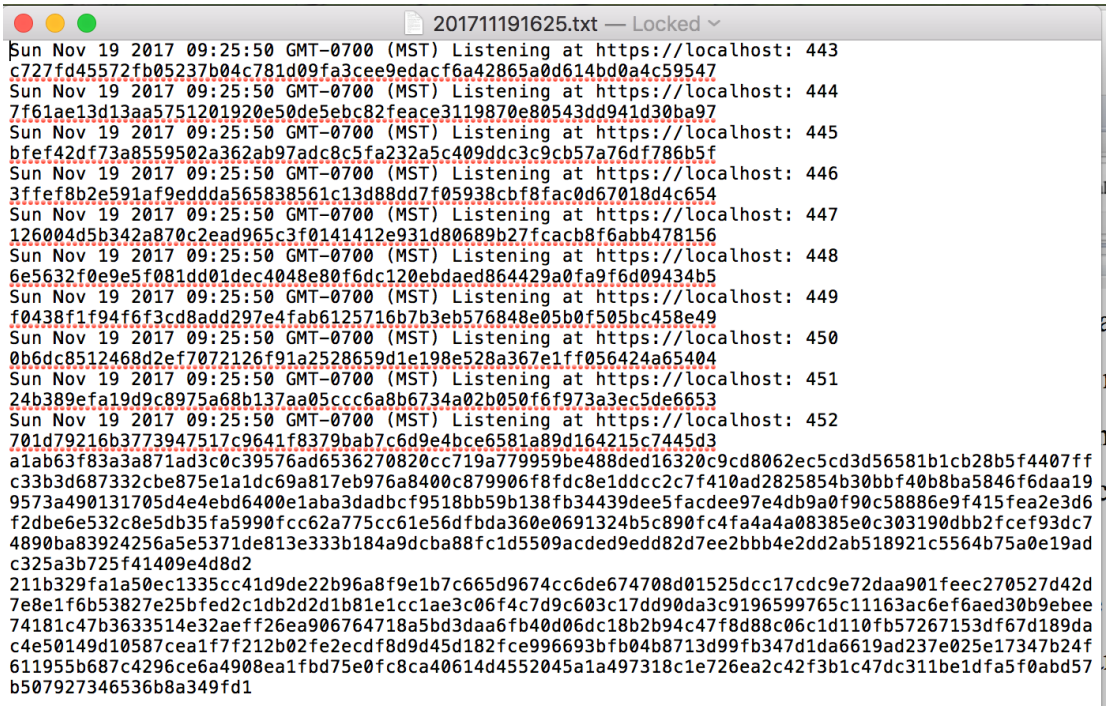
Fuente: elaboración propia.

2) Cuando se crea un nuevo archivo se firma el archivo anterior mediante un algoritmo de firma digital para asegurar la integridad de estos durante su almacenamiento y trayecto, pues son enviados a un servidor externo para su respaldo y aseguramiento. Además de asegurar la integridad de cada archivo también entre los archivos se forma una cadena de integridad mediante el cálculo de un *hash* de la concatenación del primer *log* con el archivo de *logs* anterior.

   La cadena de integridad se implementa con el propósito de detectar el punto de quiebre, es decir, donde se presentaron modificaciones. Además de la firma para proteger la integridad del archivo se aplica una segunda firma para marcar el archivo como el último. La estructura de un archivo firmado se muestra en la figura 5.

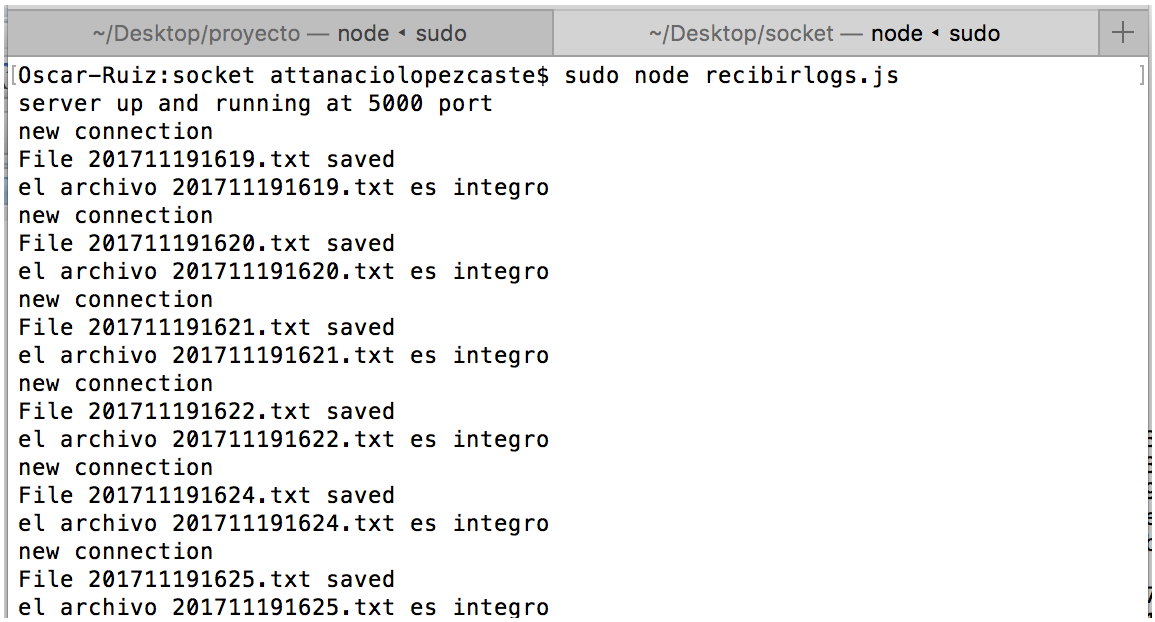
3) Una vez que el archivo es firmado se envía a otro servidor para su respaldo. El servidor de respaldo en todo momento está en la espera de recibir los archivos de *logs*. Al recibirlos verifica si el archivo no ha sufrido alguna modificación durante su trayecto. Esta validación se realiza mediante la verificación de la firma digital, ver figura 6.

Figura 5. Archivo firmado



Fuente: elaboración propia.

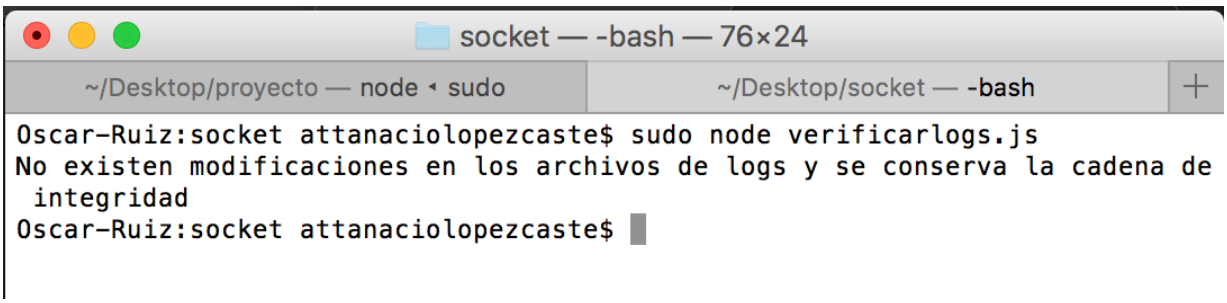
Figura 6. Servidor de respaldo



Fuente: elaboración propia.

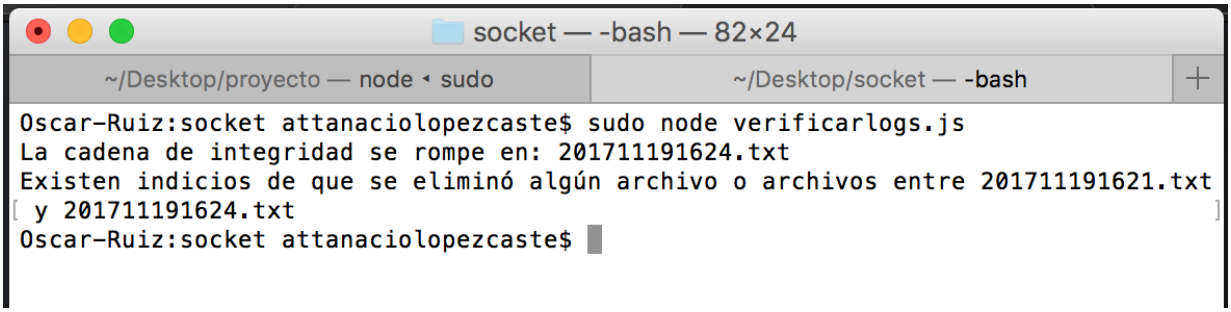
4) Una vez recibido los archivos de *logs* se puede verificar su integridad en cualquier momento. La verificación consiste en la verificación de la cadena de integridad que se forma entre cada archivo de *logs* si la cadena se conserva se muestra un mensaje, indicando que los archivos no han sufrido modificaciones y se conserva la cadena de integridad, ver figura 7. Si la cadena tiene un punto de quiebre se procede a verificar la integridad de los archivos, donde se rompe la cadena para indagar si hubo alguna modificación o se eliminó algún *log* o un archivo de *logs*, en cualquiera de los casos se muestra un mensaje indicando en qué archivo se rompe la cadena de integridad y de ser posible se muestra también el motivo, ver figura 8.

Figura 7. Verificación exitosa



Fuente: elaboración propia.

Figura 8. Verificación fallida



Fuente: elaboración propia.

El prototipo del mecanismo Medusa, como se ha explicado, tiene una serie de verificaciones que permite descubrir manipulaciones en los *logs*. De la misma forma en la que se protegen los *logs* como se ha visto en este ejemplo de implementación se pueden proteger unidades más grandes de información digital, tal como votos o cualquier otra información sensible.

Conclusiones

En este trabajo se han analizado los métodos, técnicas y mecanismos de auditoría que pueden aplicarse a los diversos sistemas de votación, iniciando con los métodos clásicos de recuentos totales y recuentos parciales a través de la selección de una muestra. También se ha descrito cómo los sistemas de verificación independiente aplicados al voto electrónico facilitan la auditoría, sin embargo, solamente para los sistemas de voto electrónico presencial.

Existen por su parte técnicas más adecuadas para realizar auditorías de sistemas de voto electrónico remoto. Algunas de ellas se basan en el análisis de *logs* o registros de eventos que se generan durante el proceso de votación. Sin embargo, la gestión y el análisis de *logs* presenta algunos retos importantes, debido a la gran cantidad de *logs* que se pueden generar en un sistema. Herramientas de análisis automático de *logs*, tal como *querifier*, facilitan la tarea del auditor para encontrar posibles manipulaciones.

En este trabajo se ha descrito un mecanismo que permite, a través de una auditoria, la detección de votos añadidos de manera ilegítima, así como la eliminación o manipulación de estos. A través de este mecanismo es posible detectar con una precisión alta en qué parte de la cadena de *logs* se ha realizado alguna manipulación, lo cual da evidencia no solo de que ha habido una manipulación, sino que se puede deducir en qué sentido se han realizado las manipulaciones detectadas. Por tanto, esta propuesta contribuye en el área de auditorías de sistemas de voto electrónico remoto, sin embargo, el mismo mecanismo puede aplicarse a otro tipo de sistemas tal como se muestra en el ejemplo de implementación del mecanismo Medusa, donde se protegen los *logs* de un sistema de información de resultados electorales.

Referencias bibliográficas

Chaum, D., Essex, A., Carback, R., Clark, J., Popuveniuc, S., Sherman, A. & Vora, P. (Mayo de 2009). Scantegrity: End-to-End Voter-Verifiable Optical- Scan Voting. *IEEE Transactions on Information Forensics and Security , 4* (4), 611-627.

Chun-IFan & Wei-ZheSun (November 2008). An efficient multi-receipt mechanism for uncoercible anonymous electronic voting. *Mathematical and Computer Modelling, 48*, 1611-1627.

Alcaraz, J., Aragon, A., Brienen, R., Fiallo, L., Friguls, A. & Gomez, J. (2012). *Final report of the Miami-Dade County grand jury: Report 2012.* Miami Dade County grand jury, Miami, USA.

Adida, B. (2006). Helios: Web-based Open-Audit Voting. *17th USENIX Security Simposium.* Vancouver, Canada.

Bellare, M. & Yee, B. (1997). *Forward integrity for secure audit logs.* University of California at San Diego, Dept. of Computer Science & Engineering.

Bernhard, M., Benaloh, J., Halderman, J. A., Rivest, R. L., A. Ryan, P. Y., Stark, P. B., . . . Wallach, D. S. (2017). Public Evidence from Secret Ballots. *2nd Joint International Conference on Electronic Voting E-VOTE-ID 2017*, (p. 419). Lochau/Bregenz.

Bretschneider, J., Flaherty, S., Goodman, S., Halvorson, M., Jinston, R. & Linderman, M. (2012). *Risk-Limiting Post-Election Audits: Why and How.* California, USA: Risk-Limiting Audits Working Group.

Morales, V. (2009). *Seguridad en los Procesos de Voto Electrónico Remoto.* Barcelona, España: Universidad Politecnica de Cataluña.

Mus, K., Sabir Kiraz, M., Cenk, M. & Sertkaya, I. (December 2016). Estonian Voting Verification Mechanism Revisited. *CoRR, abs/1612.00668*.

Riera, A. & Puiggali, J. (2004). *Pnyx Software Requirements Document.* Scytl.

Rivest, R. (November 2006). *Audit On Estimating the Size of a Statistical.* Recuperado de MIT Computer Science & Artificial Intelligence Lab: http://people.csail.mit.edu/rivest/Rivest-OnEstimatingTheSizeOfAStatisticalAudit.pdf

Rivest, R. (January 2017). *ClipAudit: A Simple Risk-Limiting Post-Election Audit.* Recupedado de MIT Computer Science & Artificial Intelligence Lab: https://people.csail.mit.edu/rivest/pubs/Riv17b.pdf

Schneier, B. & Kelsey, J. (1998). Cryptographic Support for Secure Logs on Untrusted Machines. *7th Conference on USENIX Security Simposium.* San Antonio, TX: USA.

Sandler, D., Derr, K., Crosby, S. & Wallach, D. (2008). Finding the Evidence in Tamper-evident Logs. *3rd International Workshop on Systematic Approaches to Digital Forensics Engineering.*

Stanislevic, H. (2006). *Random auditing of e-voting systems: How much is enough?* National Coalition for Election Integrity, E-Voter Education Project.

Stark, P. B. (2008). Conservative Statistical Post-Election Audits. *The Annals of Applied Statistics, 2* (2), 550-581.

Stark, P. B. (March 2010). *Why small audit batches are more efficient: two heuristic explanations.* Recuperado de University of California, Berkeley - Department of Statistics: https://www.stat.berkeley.edu/~stark/Preprints/smallBatchHeuristics10.htm

Stark, P. B. (2015). *Tools for comparison risk-limiting election audits.* Recuperado de University of California: https://www.stat.berkeley.edu/~stark/Vote/auditTools.htm

Verified Voting (2017). *State Audit Laws Searchable Database*. Recuperado de State Audit Laws Searchable Database: https://www.verifiedvoting.org/state-audit-laws/

VVSG (December 2005). *Election Assistance Commision.* Recuperado de Election Assistance Commision: https://www.eac.gov/assets/1/28/VVSG.1.0_Volume_1.PDF

Yasinsac, A. & Bishop, M. (Mayo de 2008). The Dynamics of Counting and Recounting Votes. *IEEE Security and Privacy, 6* (3), 22-29.

Reconocimiento

Los autores reconocen el apoyo del CONACYT, pues este trabajo está soportado en parte por los proyectos: Laboratorio Nacional de Tecnologías de Información – LANTI sede UACJ, y Elections Results Reliable Sample (ERRS).

* Víctor Morales Rocha cuenta con el grado de doctorado y labora en el Instituto de Ingeniería y Tecnología de la Universidad Autónoma de Ciudad Juárez (UACJ). Coordinador del Laboratorio Nacional de Tecnologías de Información, sede UACJ.

** Oscar Ruiz Hernández es estudiante de la Maestría en Cómputo Aplicado de la Universidad Autónoma de Ciudad Juárez.

*** Luis Felipe Fernández Martínez cuenta con el grado de maestría y labora en el Instituto de Ingeniería y Tecnología de la Universidad Autónoma de Ciudad Juárez. Coordinador de la Unidad de Ingeniería del Conocimiento e Ingeniería de Software, de la UACJ.

[1] Estos se clasifican como sistemas de verificación directa, sistemas de procesos separados, sistemas de testigos, o sistemas de verificación de cifrado extremo a extremo.

[2] La verificación individual se refiere a la posibilidad que tiene cada votante de verificar que su voto se registró correctamente.