Mejora de la Integridad de Datos en el Modelo OTP-Vote

Silvia Bast
Universidad Nacional de La Pampa
Email:
silviabast@exactas.unlpam.edu.ar
www.exactas.unlpam.edu.ar

Germán Montejano, Mario Berón
Universidad Nacional de San Luis
Email: {gmonte,mberon}@unsl.edu.ar
www.unsl.edu.ar

Abstract. El Modelo de OTP Vote presenta las característica de anonimato incondicional y posibilita llevar la seguridad computacional a los niveles deseados. El esquema se compone de tres procesos: preparación, desarrollo y cierre de la elección-recuento de votos. La primera etapa incluye la configuración y parametrización de la elección, la segunda abarca el proceso de emisión de votos y la tercera el cierre y recuento de sufragios. Se presenta, en este artículo, una propuesta de mejora de seguridad e integridad de los datos del modelo, que permite detectar intentos de intrusión por parte de terceros con el objetivo de modificar el contenido de los votos. Este trabajo pretende ser un aporte al incremento en el nivel de confianza del electorado, aspecto fundamental para el éxito de los sistemas de voto electrónico

Keywords: voto electrónico, integridad de datos, seguridad, funciones, bits de control

1 Introduction

La aplicación del voto electrónico presenta actualmente grandes discusiones, no sólo dentro del ámbito académico, sino también fuera de él. Quienes están en contra de su aplicación basan sus argumentos en la falta de transparencia del proceso y de los resultados. Esta postura es abonada por productos desarrollados y puestos en funcionamiento en distintas elecciones, que no cumplen con los requisitos establecidos para un sistema de eleccionario, por lo que gran parte de la sociedad percibe a tales sistemas como artefactos cerrados, no confiables, que procesan el voto de una forma oscura o desconocida y como consecuencia, opina que no deben ser usados.
Por definición, el voto electrónico es un método en el cual los votos son emitidos o tabulados por medios electrónicos. Un sistema de voto electrónico es un componente de software que mapea el procedimiento de voto electrónicamente [1] y que debe cumplir además con un conjunto de requisitos como los indicados en [2], [3], [4] y [5] que le son propios. En cuanto a los datos, los sistemas de voto electrónico deben proteger:

- La privacidad del votante (anonimato) de forma indefinida, aún después de finalizado el acto eleccionario, dado que, si algún atacante obtiene datos de los registros que mantienen la relación entre el voto y el votante podrá dedicar todo el tiempo para intentar el descifrado del mismo.
- La seguridad de los datos de los votos durante el transcurso del proceso electoral, ya que luego los resultados son de público conocimiento.

Desde el proyecto Aspectos de Seguridad en Proyectos de Software FCEyN se desarrolló el modelo OTP Vote [6] que asegura: anonimato incondicional y seguridad computacional que puede llevarse a cualquier nivel exigible durante el proceso eleccionario.

En el presente trabajo se realiza un aporte a la integridad de los datos de los votos durante el tiempo que dura el proceso electoral (10 horas), que tiene por objetivo detectar intentos de modificación fraudulenta.

El artículo está organizado como sigue: la sección 2 explica el modelo OTP Vote, la sección 3 describe la propuesta de mejora de integridad de los datos, la sección 4 presenta un caso de estudio y la sección 5 detalla las conclusiones y los trabajos futuros.

2 El Modelo OTP-Vote

Las principales características del modelo OTP Vote son:

- Uso de criptografía basada en One Time Pad (OTP, [7]) que cumple con las hipótesis y condiciones del "Secreto Perfecto" de Shannon, presentadas en [8].
- Uso de claves distribuidas que combinadas funcionan como una sola, concepto derivado de la propuesta de protocolos Broadbent y Tapp [9].
- Almacenamiento basado en el modelo de canales paralelos [10] denominado Múltiples Canales Dato Único (MCDU), que propone dividir en q canales el almacenamiento total y registra cada voto una vez en cada canal en posiciones aleatorias potencialmente distintas.

El modelo propone tres etapas, que se muestran en la Fig. 1 y se describen en las siguientes subsecciones.

Fig. 1. Proceso de OTP - Vote

Primera Etapa

La configuración o preparación de la elección incluye las siguientes actividades de configuración de los datos que se usarán durante el proceso eleccionario:

1 Definición de las dimensiones del Archivo Binario donde residirán los Votos (ABV) y Clave de Descifrado (CD) que hace uso de las fórmulas propuestas en [5].

2 Definición de las dimensiones de cada uno de los atributos que se almacenarán en las tuplas (Identificador de voto, Identificador del cargo, Identificador del candidato seleccionado, bits adicionales de control, redundancia de información que aporta a oscurecer el contenido de la tupla). Para llevar a cabo esta tarea, debe tenerse en cuenta la probabilidad de que un intruso pueda detectar una tupla válida de entre todas las tuplas posibles. Como se afirma en [6], [11] y [12], aumentando la redundancia en la cantidad de bits destinados a registrar cada atributo, la probabilidad de obtener una tupla válida de entre todas las combinaciones de valores posibles puede llevarse a cualquier valor deseado.

3 Generación de los códigos para cada uno de los atributos identificadores (Cargos, Candidatos e Identificadores de Votos). Dado que las tuplas (votos) que se almacenan usando el esquema MCDU, descripto en [11], por medio de la operación XOR [13], es necesario que los identificadores de las tablas cumplan con un conjunto de características para evitar que, a partir de votos que colisionan, el voto pueda interpretarse incorrectamente. Cada identificador (Id_i) que se agregue debe:

   a. Ser distinto a los almacenados previamente:

$$Id_i \neq Id_j \ \forall j \ 1 \leq j \leq i\text{-}1 \quad \textbf{(1)}$$

   b. El XOR del Id_i con cada uno de los identificadores que ya residen en la tabla, no debe dar como resultado alguno de los *Id* existentes:

$$Id_i \oplus Id_j \neq Id_k \ \forall j \ 1 \leq j \leq i\text{-}1, \ \forall k \ 1 \leq k \leq i\text{-}1 \quad \textbf{(2)}$$

   c. El XOR del Id_i con cada una de las combinaciones de grupos de los identificadores que ya residen en la tabla, no debe dar como resultado alguno de los *Id* existentes

   En [12] se desarrolla un algoritmo para la generación de códigos que cumplen con las características mencionadas anteriormente.

4 Configuración de la ubicación de los atributos en la tupla.
   Se denomina tupla a cada una de las filas del ABV en donde podría residir un voto. Dentro de la tupla, se guardan los identificadores de voto, cargo, candidato seleccionado, bits de control. Es posible almacenar los bits de cada uno de los atributos de manera consecutiva o dispersa dentro de la tupla, por ejemplo, un identificador de voto de 8 bits, podría estar dispuesto de la siguiente manera, los 2 primeros bits a partir de la posición 9 de la tupla, los 3 bits siguientes a partir de la posición 1 de la tupla, el siguiente bit en la posición 15 y finalmente los últimos 2 bits del identificadora partir de la posición 5.

5 Generación de las tablas: Identificadores de Votos, Cargos, Candidatos, Atributos, Ubicaciones, Votos Planos. En la Fig. 2 pueden observarse las tablas mencionadas con sus atributos.

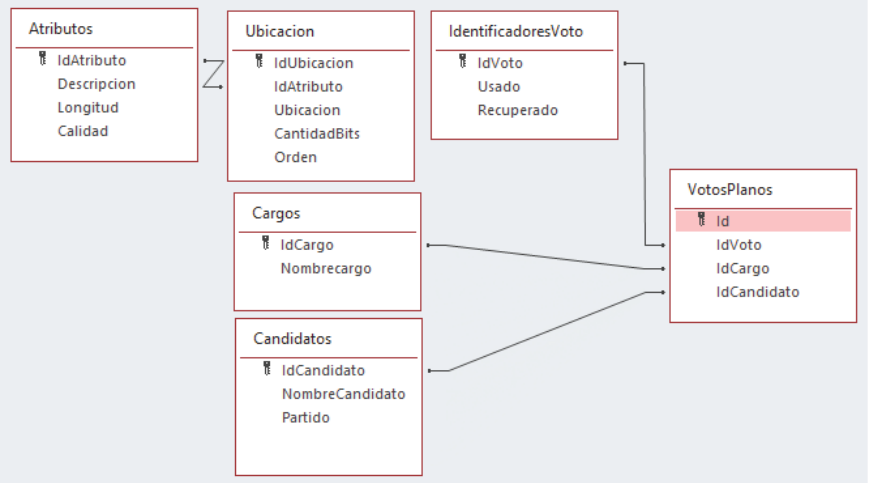

Fig. 2. Tablas de OTP - Vote

6 Finalmente, cómo última actividad de esta etapa cada autoridad aporta una clave para el ABV y una clave para CD que inicializarán ambos archivos binarios.

Segunda Etapa

Consiste en dos grandes actividades, bien diferenciadas y que se llevan a cabo por separado con el objetivo de que no se registre relación alguna entre el votante y el identificador de voto.

1. *Autenticación del elector:* se verifica que la persona que va a emitir el voto aparezca en el padrón electoral. El modelo propone que el usuario se registre en el lugar de la elección, en presencia de una autoridad de la Junta Electoral.
2. *Emisión del voto:* para el almacenamiento de los Votos se sigue el esquema MCDU. Para cada voto:
   a. El sistema selecciona aleatoriamente de la tabla IdentificadoresVoto, un Id y en caso de no estar usado, lo marca como tal. Si ya fue utilizado anteriormente repite el proceso hasta encontrar uno disponible.
   b. El sistema genera una Clave OTP ClaveVotov (CV_v) del mismo tamaño que ABV que:
      - Aportará a la CD Final de los votos, mediante operaciones XOR.
      - Cifrará la información del voto.
   c. El elector genera su voto, seleccionado el cargo y el candidato que elige para el mismo, generándose entonces $CadenadeVoto_v$, que luego se combinará con la CV_v para producir la Contribución Final de Voto (CFV).
      El aporte de la clave de voto a la CD se lleva a cabo a través de la siguiente operación:

$$CD = CD \oplus CV_v \ \forall v \ \ 1 \leq v \leq N \quad \textbf{(3)}$$

El sistema genera una cadena de *TBslot* bits $ContribucionInicial_v$ (CI_v,) con todos sus elementos en cero. Produce además un conjunto de números aleatorios $CjtoQ = \{q_i\}$ para cada uno de los Q canales, donde q_i representa el lugar donde se almacenará el voto en el canal i-ésimo.
Se realiza el XOR de la $CadenaDeVoto_v$ con los slots que corresponden a los qi de la CI_v. Esto es:

$$Contribución_{vi} = CadenadeVoto_v \oplus CI_{vi} \ \forall i \in CjtoQ \quad \textbf{(4)}$$

Finalmente:

$$CFV_v = Contribución_v \oplus CV_v \qquad ABV = ABV \oplus CFV_v \quad \textbf{(5)}$$

Tercera Etapa

Para completar la etapa de Cierre de la Elección y Recuento de Votos se requiere la intervención de las Autoridades de la Junta Electoral. El proceso de descifrado de los votos consta de tres subprocesos:

1. Aplicación de las mismas claves que las Autoridades usaron en la etapa inicial tanto al ABV como a CD.
2. XOR entre el ABV y la CD resultantes de los pasos anteriores que genera el Archivo Binario de Votos Descifrado (ABVD).

$$\text{ABVD} = \text{ABV} \oplus CD \quad \textbf{(6)}$$

3. Se procede a la recuperación de los votos [6] generando el Archivo Binario de Votos Descifrado Versión 2 (ABVDv2) que incluye dos bits más por cada tupla con el siguiente significado: (00) tupla no revisada, (01) tupla de ceros, (10) colisión y (11) voto válido y revisa cada una de las tuplas.

3 Propuesta de Mejora de Integridad de Datos de las Tuplas

Como se ha mencionado previamente, en los sistemas de voto electrónico, debe preservarse la seguridad de los datos de los votos durante las 10 horas en las que transcurre el proceso electoral. Para fortalecer esta característica y aumentar la confianza del electorado se realiza un aporte que apunta a mejorar la integridad de los sufragios. A continuación, se especifican para cada una de las etapas, las actividades que se ven afectadas por la propuesta.

Etapa 1

Una de las actividades de esta etapa, es la configuración de la ubicación de los atributos de las tuplas que se almacenan en el archivo ABV. Los atributos de las tuplas pueden residir de forma consecutiva o alternada dentro de la misma. Por ejemplo:

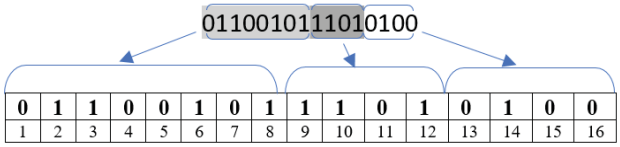

Fig. 3. Tupla con atributos en forma consecutiva

En la Fig. 3, se muestra una tupla con 8 bits para el Identificador de Voto, 4 para el Identificador de Cargo y 4 para el Identificador de Candidato, guardados de forma consecutiva. OTP Vote permite también que los atributos se almacenen en forma dispersa dentro de la tupla. A modo de ejemplo, en la Fig. 4 se muestra el Identificador de Candidato almacenado de forma dispersa.

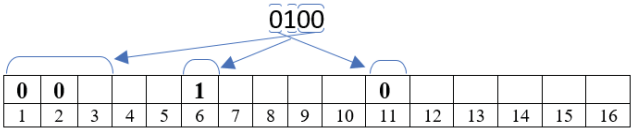

Fig. 4. Atributo Candidato almacenado en forma dispersa

Para fortalecer la integridad de los datos y de esta manera evitar intentos de modificación fraudulentos de los mismos, se propone agregar para cada atributo individual uno o más conjuntos de bits de control (BC), como así también BC a nivel de la tupla completa. Cada BC se calcula como resultado de una función *f(x)*. Dado que los mismos forman parte de la tupla, también pueden almacenarse de forma dispersa y deben cumplir con las mismas características que los demás atributos de datos. Sintéticamente y teniendo en cuenta las tablas de la Fig.2:

Sea LA_i, de longitud A_i

1. La suma de la cantidad de bits consecutivos ocupados de cada grupo de la tabla Ubicaciones para el atributo A_i debe ser igual a LA_i de la tabla Atributos

$$\sum_{j=1}^{Cantidad\ de\ Grupos} A_i LBC_j = LA_i \tag{7}$$

Donde

```
Cantidad de Grupos = SELECT
Count(Ubicaciones.IdUbicacion) AS CuentaDeIdUbicacion
FROM Ubicaciones
GROUP BY Ubicaciones.IdAtributo
HAVING (Ubicaciones.IdAtributo=Ai);

AiLBCj = SELECT Ubicaciones.CantidadBits,
FROM Ubicaciones WHERE (Ubicaciones.IdAtributo=Ai) AND
(Ubicaciones.IdUbicacion)=j);
```

2. Para todo grupo de tuplas de la tabla Ubicaciones que corresponden a un atributo A_i, debe cumplirse que la intersección de cada una de las instancias que indican bits consecutivos, con todas las demás, devuelva como resultado el conjunto vacío, esto es:

$$\bigcap_{j=1}^{Cantidad\ de\ Grupos} A_i BC_j = \emptyset \quad \textbf{(8)}$$

Donde
$A_iBC_j = \{A_iBC_k, A_iBC_{k+1}, .. A_iBC_{k+d}\}$, el conjunto formado por los bits consecutivos correspondientes al atributo $A_{i,,}$ en la instancia j de la tabla Ubicaciones, donde:
k = Bit Inicial en la instancia j
d=Bits consecutivos en la instancia j

3. Además, debe controlarse que cada atributo tenga un conjunto propio de bits en el que se alojará y que la intersección de ese conjunto con los bits adjudicados a los demás atributos devuelva cómo resultado el conjunto vacío.

$$\bigcap_{i=1}^{Cantidad\ de\ Atributos} A_i B = \emptyset \quad \textbf{(9)}$$

Donde
Sea $A_iB = \{A_iB_j, A_iB_{j+1}, .., A_iB_{cantidad\ de\ Grupos}\}$, el conjunto formado por los subconjuntos de bits consecutivos correspondientes al atributo A_i de la tabla Ubicaciones.

Un ejemplo de la Tabla Atributos puede observarse en la Tabla 1. En la misma se detallan los nombres de los atributos, la longitud, la calidad de los mismos (Datos, BC de atributos o BC de tuplas) y, en el caso de los BC se especifica la función que se aplicará y sobre qué atributo objetivo.

Tabla 1. Ejemplo de la Tabla Atributos.

IdAtributo	Descripción	Long	Calidad	Función	Atributo Objetivo
1	IdVoto	4	Dato		
2	IdCargo	2	Dato		
3	IdCandidato	4	Dato		
4	BCIdVoto1	16	BCAtributo	f(IdVoto)	1
5	BCIdVoto1	16	BCAtributo	g(IdVoto)	1
6	BCIdCargo1	16	BCAtributo	h(IdCargo)	2
7	BCIdCandidato1	32	BCAtributo	p(idCandidato)	3
8	CITupla1	32	BCTupla	K(tupla)	tupla
9	CITupla2	32	BCTupla	m(tupla)	tupla

Etapa 2

En esta etapa, la modificación radica en que luego de que se asigna aleatoriamente un Identificador de Voto al elector, y éste elige un Candidato para el Cargo objeto de la elección, el sistema calcula los BC aplicando las funciones definidas en la tabla Atributos (Fig.2), para cada uno de los atributos y también para la tupla en general.

Etapa 3

Luego de aplicar nuevamente las claves de las autoridades electorales al ABV y a CD se genera el ABVD, que se tansforma en el AVBDv2 agregándole 2 bits extras a cada una de sus tuplas y se procede a la recuperación y generación de la tabla de Votos planos, como se detalla en [6]. El algoritmo de Recuperación se modifica incluyendo el proceso de verificación de los BC.

El proceso de *Recuperación (ABVDv2,VotosPlanos*), recibe el ABVDv2 y produce la tabla de Votos Planos que se publicará luego de terminado el acto electoral y a partir de la cual se calculará el resultado de la elección.

Este proceso recorre cada una de las tuplas del ABVDv2 y evalúa los bits adicionales que se han agregado a las mismas. En el caso de que estos bits sean (0,0) que significa que la tupla aún no fue revisada o (1,0) que indica que la tupla es el resultado de una colisión, se procede a la revisión de la tupla.

El primer paso de la revisión consiste en aplicar a la tupla la función *TupladeCeros(tupla*) que devuelve verdadero si la misma consiste en una secuencia de ceros, lo que indica que en este lugar no reside ningún voto y se le asigna (0,1) a los bits adicionales correspondientes.

En caso de que no sea una tupla de ceros, se recupera el voto que está almacenado mediante el proceso *ObtenerVoto(tupla, voto),* que obtiene los datos de una tupla y los convierte en el formato esperado para un voto de acuerdo a la configuración que se estableció para la tupla en la Etapa 1.

Luego se aplica la función *VotoValido(voto)* que, dado un voto recuperado por el proceso anterior, verifica si los datos son correctos o son el resultado de una colisión. Para ello debe chequear los datos almacenados en las tablas de la Fig. 2. Si la función devuelve falso, significa que el voto es inválido, se le asigna (1,0) a los bits adicionales indicando que fue el resultado de una colisión. En caso de que sea un voto válido (con atributos correctos: Identificador de voto, identificador de cargo e Identificador de candidato), se aplica la función *VerificarBC (voto*) que vuelve a calcular las funciones para cada uno de los BC de los atributos y de la tupla general y compara los valores resultantes con los almacenados en las tuplas. Si la función devuelve Verdadero, se almacena el voto en la tabla de votos planos y se asigna (1,1) a los bits adicionales.

4 Caso de Estudio

A continuación, se especifica un caso de estudio en el que puede observarse la detección de una alteración de los datos. Se detallan los datos de las tablas de Identificadores de Votos, Cargos y Candidatos, y el ABVDv2. Para facilitar la interpretación se destinarán los primeros 8 bits de las tuplas al Identificador de Voto, los 4 siguientes al identificador de Cargo, los 4 siguientes al Identificador de Candidato y los 4 restantes al almacenamiento del BC sobre el atributo Id.Candidato.

	Archivo Binario de Votos Descifrado V2		
	Archivo Binario de Votos Descifrado	Bit Adic1	Bit Adic2
1	00101000 0110 0100 0001 1000 1001	1	1
2	10001101 0000 0010 0000 0000 0101	1	0
3	10100100 0110 0101 0001 1000 1001	1	0
4	00000000 0000 0000 0000 0000 0000	0	0
5	00000000 0000 0000 0000 0000 0000	0	0
6	00000000 0000 0000 0000 0000 0000	0	0
7	00000000 0000 0000 0000 0000 0000	0	0
8	00000000 0000 0000 0000 0000 0000	0	0
9	11011000 0110 0101 0010 1110 1001	0	0
10	00000000 0000 0000 0000 0000 0000	0	0
11	00000000 0000 0000 0000 0000 0000	0	0
12	01010101 0110 0100 0001 1001 1100	0	0

Identificadores de Voto		
Id Voto	Usado	Recuperado
00101000	Si	Si
10100100	Si	No
11011000	Si	No
01010101	Si	No

Tabla de Candidatos	
Id Candidato	Descripción
0100	A
0101	B

Tabla de Cargos	
Id Cargo	Descripción
0110	Cargo 1

Fig. 5. Datos del caso de estudio

Para el ejemplo se establece un BC para el atributo Identificador de Candidato, la función que se aplicará para la obtención del BC será *h(IdCandidato)= IdCandidato Mod 3*. En Fig. 5 se muestra el estado de ABVDv2 al momento de verificación de la tupla 3. El voto residente en la tupla 1 tiene atributos correctos y además al evaluar la función sobre el IdCandidato (0100= 4) devuelve 1, que coincide con el valor almacenado para el BC correspondiente, queda marcada como voto válido (1,1). La tupla 2, no corresponde a tupla de ceros, ni a un valor de voto válido, por lo que se marca como colisión (1,0). Al evaluar la tupla 3, devuelve valores válidos para Identificador de voto, de cargo y de candidato, pero, al aplicar la función sobre el IdCandidato(0101= 5), devuelve 2 (0010), y el valor almacenado del BC es (0001) por lo que puede detectarse una anomalía, el voto no se marca como válido y se continúa la evaluación y el proceso de recuperación de los votos. Cabe aclarar que dado que el voto se almacena en *q* canales es posible su recuperación desde alguno de los *q-1* canales restantes. [6].

En el caso de estudio puede observarse que hubo una modificación no autorizada en el Identificador del Candidato votado (en lugar de registrar el candidato A, se registró el B) que pudo detectarse gracias al uso de BC sobre ese atributo.

5 Conclusiones y Trabajos Futuros

Se presenta en este trabajo una propuesta para mejorar la integridad de datos del modelo OTP-Vote, a través del uso de bits de control (BC) para cada uno de los atributos del voto y también a nivel de tupla en general, que sumada a: el uso de claves de autoridades, las claves combinadas OTP para cada uno de los votos, la redundancia de bits usada en la configuración de los parámetros de la tupla y el proceso de recuperación en caso de colisiones, resultan un aporte integral a la seguridad de los datos usados en OTP Vote.

El modelo permite definir tantos BC como se crean necesarios para los atributos de datos, estableciendo para cada uno de ellos, una función que permita obtener un número que se almacenará en la tupla. Por otro lado, también es posible establecer BC con funciones que se aplican a la totalidad de los atributos de datos de la tupla.

Cuando el voto está en proceso de verificación y recuperación, se aplican nuevamente la/s función/es general/es a los atributos y en caso de que no devuelvan el mismo valor que está almacenado en los bits destinados al/los BC puede concluirse que hubo alguna alteración en los valores originales. Como quedó probado en el caso de estudio pudo detectarse una modificación en el dato del candidato votado, lo que podría afectar el resultado de la elección.

Esta contribución presenta un importante aporte a la integridad de los datos de los votos, reforzando el control sobre la modificación de los mismos. Además, teniendo en cuenta que se propone hacer uso de código abierto para el modelo, la mejora puede llevarse a cabo de forma sencilla, agregando al proceso una función de resultado booleano, que no afecta la legibilidad del algoritmo.

A futuro, será necesario realizar un profundo análisis sobre las funciones que resulten más apropiadas para aplicar a los atributos individuales y a la tupla en general. Además, se deberá trabajar para permitir que el modelo facilite el proceso de auditoría de terceros e incluya mejoras en la verificabilidad E2E.

References

1. Odrisek, B.“E-Voting Security Study“-Communications-Electronics Security Group, X/8833/4600/6/21, (Copyright The Crown) Issue 1.2 31 United Kingdom, 2002
2. Epstein, J. “Electronic Voting”, Cyber Defense
3. Kazi Md. Alam Rokibul; Tamura, Shinsuke, “Electronic Voting - Scopes and Limitations” IEEE/OSA/IAPR International Conference on Infonnatics, Electronics & Vision.
4. Prince, A. “Consideraciones, aportes y experiencias para el Voto electrónico en Argentina”. 2005.
5. van de Graaf, J., Henrich C., Müller-Quade J. ”Requirements for secure voting”. 2011.
6. Bast, S. “Confidencialidad e Integridad de Datos en Sistemas de E-Voting – Un Modelo para la Implementación Segura de un sistema de Voto Presencial” - Editorial Académica Española. https://www.eae-publishing.com-ISBN 978-3-639-53793-2. 2017.
7. Nagaraj N., Vaidya V., Vaidya P: “Revisiting the one-time pad,” International Journal of Network Security, vol. 6, no. 1, pp. 94-102, 2008.
8. Shannon,C. “Communication Theory of Secrecy Systems”-Bell System Tec Journal-1949
9. Broadbent A., Tapp A.: "Information-Theoretically Secure Voting Without an Honest Majority". In Proceedings of the IAVoSS Workshop On Trustworthy Elections (2008).
10. Garcia, P. “Una Optimización para el Protocolo Non Interactive Dining Cryptographers” - Editorial Académica Española (https://www.eae-publishing.com/ - ISBN-13: 978-3-639-85270-7. ISBN-10: 3639852702. EAN: 9783639852707 – 2017.
11. Bast, Silvia Gabriela; García, Pablo Marcelo; Montejano, Germán Antonio. “Modelo de Datos del Sistema de Voto Electrónico Presencial OTP-Vote”. Memorias de las 46 Jornadas Argentinas de Informática e Investigación Operativa (JAIIO). SIE 2017 Simposio de Informática en el Estado. 4 al 8 de setiembre de 2017 UTN. Córdoba. Arg.. ISSN: 2451-7534
12. Bast, Silvia, García Pablo, Montejano Germán: ”Generación de Códigos para OTP - Vote”. Actas de 5to Congreso Nacional de Ingeniería Informática / Sistemas de Información. Aspectos Legales y Profesionales y Seguridad Informática” ISSN: 2347-0372 © CONAIISI 2017. Pág 12-22. Noviembre de 2017. Santa Fe.
13. Murdocca M., Heuring V. “Principles of Computer Architecture. Appendix A: Digital Logic”. Editor: Addison Wesley; Edición: US ed (29 de noviembre de 1999) Idioma: Inglés - ISBN-10: 0201436647 - ISBN-13: 978-0201436648.