



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

ELECCIÓN DE VOCALES DE MESA CON ALEATORIEDAD VERIFICABLE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

FRANCO ANÍBAL PINO CÓRDOVA

PROFESOR GUÍA:
ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
FEDERICO OLMEDO BERÓN
TOMÁS BARROS ARANCIBIA

Este trabajo ha sido parcialmente financiado por el proyecto NIST 60NANB18D215

SANTIAGO DE CHILE
2019

Resumen

En los procesos electorarios en Chile, miembros de la ciudadanía pueden tener que cumplir el rol de “vocal de mesa”, cuya función es estar a cargo de las mesas receptoras de sufragios. Los vocales de mesa son los primeros encargados de los votos durante los procesos democráticos. Por esto, es de suma importancia que se elijan de forma correcta y según como se describe en la ley chilena. Por otro lado, la misma ciudadanía frecuentemente tiene quejas respecto a la elección de vocales de mesa, por ejemplo, como ser elegido múltiples veces. Estas quejas son comúnmente dirigidas al Servicio Electoral de Chile cuando en realidad muchas debiesen ser dirigidas a la junta electoral relevante, quienes son las encargadas de designar vocales de mesa.

El proceso de designación incluye un sorteo aleatorio. Las juntas electorales dejan registros de su trabajo en actas y nóminas guardadas en libros en la oficina de alguno de sus miembros, lo que quiere decir que la información sobre cómo se eligen en definitiva los vocales está disponible solo en espacios físicos, y no es del conocimiento de todos.

Surge así la posibilidad de mejorar el proceso, modificando la selección de vocales de modo de proveer una forma confiable, que esté ampliamente disponible, para verificar su correctitud. Al mismo tiempo, podría ayudar a mejorar la situación presentar con más información sobre el proceso a quien se elija como vocal.

El trabajo de esta memoria consiste en la implementación de un prototipo de aleatoriedad verificable en el sorteo de vocales de mesa, donde la verificación pueda hacerse sin necesidad de asistir físicamente al sorteo o acceder a documentos físicos. Para lograr esto, se usa el servicio público de aleatoriedad verificable del CLCERT, y se contacta con el SERVEL y la junta electoral de Macul para obtener asistencia y retroalimentación. Se crean prototipos e idean diseños sobre: 1) cómo modificar la tecnología desarrollada por el SERVEL, que ya es utilizada por las juntas electorales, para añadir el uso de aleatoriedad verificable, y 2) cómo modificar páginas de internet del SERVEL ampliamente usadas por la ciudadanía para ver si se es vocal de mesa, para añadir información sobre la aleatoriedad verificable y ayudar a visitantes de dichas páginas a comprobar que el sorteo fue correcto.

Se establece un diseño que demuestra la posibilidad de una implementación en un ambiente real, identificando la información mínima que el SERVEL debiese guardar para habilitar la posterior verificación del sorteo, y se implementan maneras de informar a vocales de mesa al respecto y habilitar la verificación de la aleatoriedad usada.

Agradecimientos

Quiero agradecer al profesor Hevia, por darme el tema de esta memoria y apoyarme en todo el proceso. También a todos los de Random UChile, que me recibieron cálidamente al grupo. A la profesora Nancy, por su ayuda, calidez y paciencia en mi primer intento de memoria. A Sandra y Angélica por ayudarme con tantos procesos en esta carrera y responder a tantas preguntas que les hice. Y, finalmente, a mis amigos y familia, que siempre han estado ahí para elevarme los ánimos y apoyarme.

Tabla de Contenido

1. Introducción	1
1.1. Contexto	1
1.2. Objetivos	2
1.2.1. Objetivo General	2
1.2.2. Objetivos Específicos	3
1.3. Descripción General de la Solución	3
2. Marco Teórico	5
2.1. Teoría de los Procesos Eleccionarios Chilenos	5
2.1.1. Introducción a los Procesos Eleccionarios	5
2.1.2. Juntas Electorales	6
2.1.3. Selección de Vocales de Mesa, Según la Ley	6
2.2. Procesos Eleccionarios en Concreto	7
2.2.1. Sorteo de Vocales en la Actualidad	7
2.2.2. Ordenando Alfabéticamente	7
2.2.3. SIVOME: Aplicación Web para las Juntas Electorales	8
2.2.4. Verificación de Datos Electorales en la Actualidad	8
2.3. Aleatoriedad Verificable	10
2.3.1. Fuentes de Aleatoriedad	10
2.3.2. Faro de Aleatoriedad Verificable del CLCERT	11
2.3.3. Uso Actual del Faro de Aleatoriedad	13
2.4. Consideraciones sobre Cómo Elegir Números al Azar	13
3. Descripción del Problema	17
3.1. Problema y Oportunidades	17
3.2. Relevancia	17
3.3. Especificaciones de la Solución	19
3.3.1. Características Generales	19
3.3.2. Generación de Números Pseudoaleatorios	19
3.3.3. Elección de Fuente de Aleatoriedad Verificable	20
3.3.4. Sistema de Sorteo	20
3.3.5. Sistema de Verificación	21
4. Solución	23
4.1. Diseño General	23
4.2. Procedimiento del Sorteo	24

4.2.1.	Preámbulo	24
4.2.2.	Obtención de Aleatoriedad	24
4.2.3.	Procesamiento de la Aleatoriedad	25
4.2.4.	Algoritmo de Sorteo	25
4.3.	Librería de Funciones Aleatorias	26
4.3.1.	Descripción General	26
4.3.2.	Generando un Número en un Rango	28
4.4.	Fuente de Aleatoriedad Verificable	29
4.5.	Backend	30
4.5.1.	Construcción del Backend	30
4.5.2.	Modelo de la Base de Datos	30
4.5.3.	Petición de Pulso de Aleatoriedad al Faro	31
4.5.4.	Otros Servicios Web	34
4.6.	Sistema de Selección de Vocales de Mesa	35
4.6.1.	Plan de Intervención	35
4.6.2.	Nuevo Sorteo	37
4.7.	Sistema de Verificación Sorteo de Vocales de Mesa	43
4.7.1.	Plan de Intervención	43
4.7.2.	Modificaciones a Consulta SERVEL	45
4.7.3.	Página de Explicación Detallada de la Selección	47
4.7.4.	Página de Detalles Técnicos del Sorteo	54
4.7.5.	Verificando la validez del Pulso	55
5.	Validación	58
5.1.	Conformación a las Especificaciones	58
5.2.	Pruebas	59
6.	Conclusión	63
6.1.	Resumen	63
6.2.	Revisión de Objetivos	63
6.3.	Discusión de Resultados	64
6.4.	Trabajos Futuros	65
	Bibliografía	67
	Anexos	70

Índice de Ilustraciones

2.1. Sorteo en el manual del SIVOME	9
2.2. Consulta de Datos Electorales del SERVEL	10
2.3. Certificado CLCERT	12
4.1. Diagrama Interacciones	23
4.2. Mockup sorteo SIVOME	36
4.3. Diagrama Interacciones: Acercamiento sistema de selección	37
4.4. Prototipo sorteo verificable SIVOME: cargando	38
4.5. Prototipo sorteo verificable SIVOME: seleccionando	40
4.6. Prototipo sorteo verificable SIVOME: reiniciado	41
4.7. Prototipo sorteo verificable SIVOME: confirmando finalización	42
4.8. Nueva consulta.servel: parte 1	44
4.9. Nueva consulta.servel: parte 2	46
4.10. Detalles de selección de vocal: primera vista	48
4.11. Detalles selección: de la lista de 30 a la nómina final	49
4.12. Recreando el sorteo: demostración interactiva en estado inicial	50
4.13. Recreando el sorteo: demostración interactiva en curso	51
4.14. Recreando el sorteo: vista de resultados de demostración interactiva	51
4.15. Generadores y semillas: sección interactiva sobre generadores pseudoaleatorios	52
4.16. Origen semilla: sección interactiva sobre funciones de hash	54
4.17. Ejemplo Completo: integración interactiva de todo	55
4.18. Detalles técnicos: primera mitad de los detalles técnicos	56
4.19. Verificador: éxito	56
5.1. Frecuencias relativas sorteo, usando dos generadores	61
5.2. Frecuencias relativas primer número seleccionado	62
1. Modal de confirmación de sorteo del manual del SIVOME	70
2. Prototipo sorteo verificable SIVOME: confirmando finalización	70
3. Detalles selección: lista marcada	71
4. Recreando el sorteo: explicación de reconstrucción de selección	71
5. Generadores y semillas: explicación generador y semillas	72
6. Generadores y semillas: generador interactivo, estado inicial	72
7. Generadores y semillas: parte final	72
8. Origen semilla: explicación sobre cómo se obtiene la semilla	73
9. Origen semilla: individualización de la semilla	73
10. Ejemplo Completo: introducción ejemplo completo	74

11. Verificador: cargando	74
12. Verificador: inválido	75
13. Detalles técnicos del sorteo: página completa	76

Capítulo 1

Introducción

1.1. Contexto

En escenarios donde se usa aleatoriedad para tomar decisiones, puede ser importante que el sistema sea transparente y confiable para aquellas personas afectadas por esas decisiones. Sin embargo, hay procesos para los cuales no siempre es obvio o si quiera verificable que efectivamente se usó aleatoriedad en la decisión en lugar de juicio humano.

Un ejemplo concreto puede encontrarse en los procesos electorarios de Chile, donde existe el rol de “vocal de mesa”. Los vocales de mesa son aquellas personas que integran a las mesas receptoras de sufragios, reciben los votos de los electores y se encargan de realizar el primer escrutinio de los mismos, además de otras funciones según está escrito en la Ley 18700 [1]. Para cada mesa receptora, los vocales de mesa son elegidos por las juntas electorales. Para el caso de mesas nuevas o que necesiten nuevos vocales, las juntas primero elaboran una lista de candidatos compuesta de 30 personas, a partir de un universo compuesto por todo ciudadano que pertenezca al padrón electoral de esa mesa y que, según las definiciones dadas en la Ley 18700, pueda ejercer el rol de vocal. Una vez se tenga esa lista, los nombres se ordenan alfabéticamente y se le asigna a cada uno un número del 1 al 30. Luego de que la junta electoral haya confeccionado todas las nóminas necesarias, se da lugar a un sorteo en sesión pública, mediante el cual se obtienen 10 números al azar entre 1 y 30, correspondiendo estos a los números asignados a los nombres en las nóminas antes mencionadas. Los primeros 5 de esos números indican quienes desempeñarán el papel de vocales de mesa, y los 5 restantes indican a sus reemplazantes.

Los sorteos son muy comúnmente realizados por medio de “tómbolas”, el trigésimo día antes de la elección o plebiscito. Sin embargo, el hecho de que estos sorteos existan y sean públicos no es necesariamente de conocimiento común, y aunque lo fuese, el requisito de “sesión pública” no obliga a que sea masivo. De hecho, el espacio en estas sesiones está limitado pues, por ley, se realizan en las oficinas de quien cumpla el papel de secretario de la junta electoral [1, Artículo 46].

Actualmente, el Servicio Electoral de Chile, o SERVEL, provee a las juntas electorales

de una aplicación con la que ellas podrían seleccionar a los vocales de mesa y miembros del colegio escrutador, y también registrar y entregar la información de la selección al SERVEL. Usualmente la selección misma se realiza por los medios físicos antes descritos, en los sorteos públicos, por lo que la aplicación en la práctica se usa solo para el registro de los resultados. Esta aplicación se conoce como SIVOME (Sistema de Vocales de Mesa y Miembros de Colegios Escrutadores).

Cuando las personas son (o no) elegidas como vocales de mesa, puede haber una falta de entendimiento y una desconfianza sobre cómo se ha realizado el proceso. Frecuentemente vocales cuestionan la aparente arbitrariedad de su elección llegando a dudar de la aleatoriedad de ésta.

En este trabajo de título lo que se pretende es implementar un sistema donde esos 10 números aleatorios antes mencionados, a través de los cuales se eligen vocales de mesa, sean obtenidos por medio de aleatoriedad verificable, vía escoger números al azar donde luego se pueda certificar y demostrar que aquellos números fueron efectivamente elegidos fuera del control y juicio de cualquier persona. Esto implica no solo posiblemente modificar cómo las juntas electorales realizan sus sorteos, sino que también asegurar que haya una forma en que personas externas puedan verificarlos independientemente.

El desafío es implementar algo que sea cómodo para las juntas electorales, integrable al ambiente real y que sea convincente para la ciudadanía.

Se especifican más los problemas y oportunidades a abordar, así como la relevancia de hacerlo, en el Capítulo 3.

1.2. Objetivos

1.2.1. Objetivo General

El objetivo general es implementar un sistema de aleatoriedad verificable en el proceso de elección de vocales de mesa. El sistema debe permitir a las juntas electorales generar fácilmente los números al azar necesarios para elegir de entre los candidatos a vocales, pero también debe ser tal que cualquier persona, en particular quienes fuesen seleccionadas como vocales de mesa, puedan verificar que los números usados por las juntas electorales fueron efectivamente aleatorios.

Considerando que no se ha formalizado una cooperación directa con el SERVEL, implicando esto que no se está implementando en los sistemas reales, lo que se busca es implementar prototipos y llegar a diseños que demuestren estas cualidades, así como las que se especifican en la sección de Objetivos Específicos.

1.2.2. Objetivos Específicos

1. Diseñar una forma de conectar el sistema actualmente usado por las juntas electorales, incluidos interfaz y backend, a sistemas de generación de números verificablemente aleatorios.
2. Diseñar una interfaz que permita a las juntas electorales usar dichos sistemas para seleccionar vocales de mesa. La interfaz debe ser lo más simple posible, a fin de suavizar el impacto de las modificaciones en el software que usan las juntas electorales y minimizar la reticencia al cambio de su parte.
3. Diseñar un proceso que, luego de realizada una elección de vocales de mesa, permita a cualquier persona ver si fue seleccionada como vocal de mesa o no, y si lo fue, ver si fue elegida mediante sistemas de aleatoriedad verificable y, finalmente, si lo fue, permita a esta persona verificar por sí misma la aleatoriedad del sorteo. Todo esto debe ser lo más simple y bien explicado posible, ya que debe ser accesible para la mayor cantidad de ciudadanos chilenos posible.

1.3. Descripción General de la Solución

Durante el trabajo de memoria se esperaba poder trabajar directamente cooperando con el SERVEL, para poder diseñar y crear prototipos de la solución de forma que fuese lo más cercana posible a un sistema completamente funcional en un ambiente real. Sin embargo, no se han podido concretar ciertos acuerdos formales necesarios para lograrlo, por lo que en lugar de esto se opta por trabajar en los objetivos de forma más directa y se busca encontrar qué elementos deben tenerse para lograr la selección de vocales de mesa con aleatoriedad verificable.

Lo hecho consistió principalmente en trabajar en el diseño de un sistema de selección verificable y en el sistema de verificación correspondiente, haciendo uso de una fuente pública de aleatoriedad verificable. Adicionalmente, se necesita elegir un generador de números pseudoaleatorios apropiado, el que se usa tanto en la selección de vocales de mesa, como durante el proceso de verificación (para reconstruir la selección), por lo que el trabajo también consideró estudiar generadores, buscar implementaciones/librerías existentes y decidir qué usar o implementar.

El servicio de aleatoriedad pública que se utiliza es el faro de aleatoriedad desarrollado por el Laboratorio de Seguridad Computacional y Criptografía Aplicada (o CLCERT) [2], el cual emite un “pulso de aleatoriedad” verificable cada minuto. Se hace uso también de los scripts de verificación que el CLCERT ha hecho públicos en GitHub.

Para la selección de vocales de mesa verificable, se hace un prototipo basado en la página del SIVOME donde se registran los resultados del sorteo de vocales de mesa. El prototipo permite usar un pulso de aleatoriedad para elegir los 10 números. Para esto se conecta a un backend hecho en Python con Flask, que usa la API del faro de aleatoriedad para obtener un pulso y registrarlo en la base de datos antes de pasarlo hacia el frontend. El backend

también se hace como parte del trabajo para identificar el mínimo de datos necesarios que se necesitaría tener en una implementación real, pero que con toda probabilidad el SERVEL actualmente no modela en sus bases de datos.

El trabajo hecho para el sistema de verificación consiste en crear prototipos de qué cambios pueden realizarse a la página de Consulta de Datos Electorales del SERVEL con el fin de que en ésta se les informe a vocales de mesa sobre la elección verificable. Esto incluye, además de modificar la página ya existente, crear otra donde haya una explicación más a fondo y avanzada donde pueda empezar a verse una forma más real de verificación (incluyendo reconstruir la manera en la que se seleccionan los números, paso a paso). Adicionalmente, la modificación de la página de Consulta de Datos Electorales incluye añadir ahí otra información relevante a vocales de mesa, como información sobre sus juntas electorales, y detalles de cómo excusarse.

Finalmente, se crea también una página donde se puede verificar la correctitud de pulsos de aleatoriedad haciendo uso de los scripts que el CLCERT ha hecho disponibles para este fin. El backend para esto también es hecho a través de Python y Flask, aprovechando que dichos scripts están hechos en Python, si bien se modifican un poco para convertirlos en una función que se pueda importar, llamar y que retorne resultados (o errores) en un formato conveniente para luego pasarlos al frontend (en lugar de scripts cuyos resultados solo se imprimen directamente en la consola).

Como parte del trabajo de memoria, se implementó también un generador de números pseudoaleatorios para JavaScript, sobre una implementación existente del cifrador de flujo ChaCha20. Esto incluyó implementar varias otras funciones relacionadas al uso de azar, al punto en que se llegó a una primera versión de una librería de tales funciones (además de otras utilidades) basada en ChaCha20, la cual se dejó en un repositorio público en GitHub [3]. Experimentalmente, en el proyecto “Random UChile” del CLCERT ya se está probando usar esta librería, e incluso ya ha sido portada a Python[4].

Para la selección selección de 10 índices entre 1 y 30, se implementó un algoritmo de sorteo usando el generador de números basado en ChaCha20 antes mencionado.

Durante el trabajo de memoria también se consideró la posibilidad de realizar verificación masiva de los datos de vocales de mesa. Esto sería, implementar una manera de hacer los procesos de verificación para todas las circunscripciones donde se use aleatoriedad verificable, pues lo implementado muestra extensibilidad a tal posibilidad. Esto hace el supuesto de que los datos de las nóminas pueden estar disponibles de forma masiva a terceras partes, que puede ser una suposición demasiado fuerte (actualmente los datos si bien están disponibles con fácil acceso, las páginas de internet están hechas para ver datos de personas individuales, y usan sistemas “reCAPTCHA”, que están hechos para limitar el acceso a usuarios humanos). Además, esta idea se tendría que construir usando el resto del trabajo realizado en esta memoria como base, por lo que de todos modos necesitaría ser implementada posteriormente. Por todo esto, se deja registro aquí de esta propuesta, pero se deja de lado durante el trabajo de memoria.

Capítulo 2

Marco Teórico

2.1. Teoría de los Procesos Eleccionarios Chilenos

2.1.1. Introducción a los Procesos Eleccionarios

Para los propósitos de este trabajo de memoria, es necesario conocer algunos puntos sobre cómo funciona la democracia chilena. Sin embargo, es un tema complejo y explicar todo sobre estos procesos se escapa del alcance de este informe, por lo que simplemente se dan las definiciones que permiten entender las ideas más relacionadas a lo que es el “sorteo de selección de vocales de mesa”.

En primer lugar, sobre el sistema político chileno a grandes rasgos basta mencionar que es una república democrática presidencialista, en la cual se llevan a cabo elecciones donde la ciudadanía votará para elegir a sus presidentes (y otros varios cargos de la jerarquía gubernamental), además de haber la posibilidad de plebiscitos donde también existirán votaciones que llevar a cabo.

Con respecto a los procesos de votación y escrutinio de los votos, que encierran el contexto relevante para esta memoria, la forma en la que estos funcionan está dictada y demarcada en la ley chilena, más específicamente en la Ley Orgánica Constitucional Sobre Votaciones Populares y Escrutinios [1].

Para poder realizar estas votaciones existe lo que se llaman “mesas receptoras de sufragios” [1, Título I, párrafo 7º], cuyos propósitos más importantes son recibir los votos y realizar luego un escrutinio de los mismos. Una mesa receptora tiene normalmente cinco integrantes, y a quienes integren una mesa receptora se les conoce como “vocales de mesa”. La forma en el que se designan estos vocales es uno de los focos de este trabajo de memoria.

2.1.2. Juntas Electorales

Las juntas electorales son las autoridades administrativas encargadas de designar a los integrantes de los colegios escrutadores, a los delegados de las oficinas electorales y a los vocales de mesa. Cada provincia de Chile tiene una junta electoral y cada una de estas juntas está compuesta por tres personas, quienes, por ley, son aquellas personas en ciertos cargos de la administración judicial de la provincia. Por ejemplo, de haber, el conservador de bienes raíces de la provincia es el secretario de la junta electoral. Los detalles de la conformación y rol de la junta electoral se escriben en el Título XII de la Ley 18700 [1] (y también a partir del Artículo 210 para las juntas electorales en el extranjero).

Cada junta electoral vela por una o más circunscripciones electorales, que son subdivisiones territoriales, que a su vez pueden verse como agrupaciones de locales de votación y de mesas receptoras. Más concretamente, a la fecha de las elecciones presidenciales chilenas del 2017, contando tanto dentro de Chile como en el extranjero, había más de 200 juntas electorales, designando más de 200 mil vocales de mesa, para más de 42 mil mesas receptoras de sufragios [5].

2.1.3. Selección de Vocales de Mesa, Según la Ley

El proceso de selección de vocales de mesa está definido en la Ley 18700 [1, Título I, párrafo 8º], y los procedimientos descritos en aquella ley son ejecutados por las juntas electorales con ayuda del SERVEL.

Para cada mesa, la junta electoral debe confeccionar una nómina con 30 candidatos a vocales de mesa, elegidos del padrón electoral de la mesa (que indica quiénes emiten votos en esa mesa). Los 30 puestos a llenar se distribuyen equitativamente entre integrantes de la junta electoral, es decir, si son tres integrantes, cada uno elige 10 personas. La selección de candidatos se hace según a quién los integrantes de la junta consideren más idóneos para el trabajo, entendiendo además que por ley se debe excluir de la selección a toda persona que cumpla ciertas características (por ejemplo, si el proceso es una elección presidencial, no pueden ser vocales quienes estén de candidatos a presidente, ni tampoco sus parientes más cercanos). Esta nómina luego se ordena alfabéticamente para asignar un índice a cada persona. Finalmente, todas las nóminas ordenadas son firmadas por los integrantes de la junta electoral y se juntan en un libro que se entenderá como parte del acta del proceso, el cual luego será público, mantenido bajo la custodia del secretario de la junta.

Luego de haber creado y ordenado estas listas para cada una de las mesas receptoras de las que la junta electoral esté encargada, se pasa a realizar un sorteo para seleccionar de entre los candidatos.

El sorteo es elegir 10 números al azar (secuencialmente y sin repetición) entre 1 y 30, los cuales representarán índices de candidatos para las listas de todas las mesas, de todas las circunscripciones bajo el cargo de la junta electoral. Los números se usan en el orden en que fueron generados, según se necesiten para cada masa, generalmente con los primeros 5 números designando vocales titulares y los últimos 5 a suplentes, sin embargo no todas las

mesas necesitan 5 vocales nuevos, en cuyo caso no se usarán todos los números para esa mesa.

Se debe realizar el sorteo en una sesión pública, a las catorce horas del trigésimo día anterior a la fecha de la elección o plebiscito, en la oficina de quien cumpla el papel de secretario de la junta electoral [1, Artículo 46].

2.2. Procesos Eleccionarios en Concreto

2.2.1. Sorteo de Vocales en la Actualidad

Lo visto sobre cómo sortear en las secciones anteriores corresponde a las definiciones dadas por ley, que todas las juntas electorales deben seguir, sin embargo el método exacto para seguir la letra de la ley depende de cada junta electoral.

Con respecto a cómo se eligen los vocales de mesa en la actualidad, un método popular es el uso de una tómbola física, donde los números se van eligiendo uno a uno en presencia y bajo el escrutinio de quienes fuesen los asistentes al sorteo. Cabe mencionar que un aparato físico como lo es una tómbola puede también ser manipulado [6].

2.2.2. Ordenando Alfabéticamente

Un punto a tomar en cuenta sobre las secciones anteriores, es que la ley citada solo menciona que las nóminas se deben “ordenar alfabéticamente”, pero no qué significa esto.

El orden correlativo que se aplique entre letras o caracteres puede diferir dependiendo del contexto, e incluso dentro de un mismo contexto nacional el orden aplicado puede diferir. Por ejemplo, distintas personas pueden decidir ordenar puntuación y números de formas distintas en relación al alfabeto, o dar distintas prioridades a mayúsculas contra minúsculas. Esto puede ser problemático en términos de que para poder reproducir el proceso de selección de vocales, se debe aplicar el mismo algoritmo para ordenar tanto al momento de realizar el sorteo como al momento de hacer la verificación, por lo que debe existir un acuerdo entre ambos lados al respecto.

Por otro lado, existe el estándar internacional ISO 14651 [7], que especifica cómo ordenar, incluyendo dando distintos órdenes para distintos lenguajes y culturas. A su vez, hay una especificación técnica de Unicode, conocida como “Unicode Collation Algorithm” [8] que también detalla cómo ordenar alfabéticamente, se alinea con ISO 14651 y se ve implementada en librerías ampliamente portadas a través de distintos lenguajes de programación [9].

Actualmente, el SERVEL provee a las juntas electorales de una aplicación web para ayudarlas a llevar a cabo procedimientos relacionados a sus roles (se detalla más de esto en la siguiente sección). Dentro de lo que la aplicación hace, está en darle un orden alfabético a la nómina de manera automática, por lo que el orden que usan las juntas electorales, por lo

menos dentro de las que usen este software, es el dado por dicha aplicación.

2.2.3. SIVOME: Aplicación Web para las Juntas Electorales

Junto al uso de aparatos como la tómbola, las juntas electorales suelen usar una aplicación web llamada SIVOME (abreviación de “Sistema de Vocales de Mesa y Miembros de Colegios Escrutadores”). Esta aplicación es provista por el SERVEL para ayudar a las juntas con los procedimientos descritos en secciones anteriores (así como otros no descritos), y además hace de canal de comunicación entre las juntas electorales y el SERVEL.

El SIVOME es levantado durante los procesos electorarios para uso exclusivo de las juntas electorales, a quienes se les dan credenciales para ocupar la aplicación. A través de ésta, las juntas electorales reciben información sobre las personas que están bajo su jurisdicción, para que puedan confeccionar las nóminas de candidatos.

Además, gracias a la aplicación tienen un sistema con el que organizar y registrar las mesas y nóminas. De particular importancia para los propósitos de esta memoria, es que en el SIVOME también se registran los 10 números elegidos durante el sorteo, como se muestra en la figura 2.1. Puede que en la imagen del manual del SIVOME (dado que es una imagen del manual físico) no se aprecia del todo lo que está en la imagen: bajo los números hay cuadros que son campos de texto. Inmediatamente bajo la última línea celeste, en el extremo izquierdo, dice “Finalizar”, palabra la cual se muestra con un estilo claro para dar a entender que la sección para finalizar el sorteo está inhabilitada, pues aún no se ingresan los 10 números en el escenario mostrado en la imagen.

El registro del sorteo es uno de los últimos pasos de la selección de vocales de mesa en el SIVOME y solo puede realizarse una vez se hayan registrado las nóminas para todas las mesas (en caso de intentar hacer el sorteo antes, la aplicación detiene al usuario y le muestra una pantalla con las mesas pendientes). Una vez confirmados los números, el SIVOME ofrece a las juntas electorales ver y descargar varios documentos y reportes asociados a los procedimientos llevados a cabo. El uso del SIVOME implica también que información de las actas del sorteo, como por ejemplo las nóminas de candidatos, es reportada al SERVEL y por lo tanto se mantiene digitalmente en su sistema.

Cabe mencionar que el sorteo puede también deshacerse y además que hay otras etapas en el proceso de selección de vocales de mesa, como lo es el periodo de excusas donde vocales seleccionados pueden excusarse del trabajo, para lo cual el SIVOME también ofrece funcionalidades.

2.2.4. Verificación de Datos Electorales en la Actualidad

En el caso actual, se podría entender que el “sistema de verificación” se tiene por medio de asistencia el evento físico del sorteo de vocales, y por las actas que las juntas electorales deben llevar sobre la selección que se mantienen en custodia por el secretario de la junta [1,



Figura 2.1: Captura de pantalla de parte de la sección del manual del SIVOME donde se explica el registro del sorteo. Un PDF de dicho manual se obtuvo gracias al SERVEL.

Artículo 47].

Dado que dichos documentos residen en libros físicos cuyo contenido no es generalmente anunciado de formas tan públicas como la lista final de vocales, en la práctica el acceso a estas nóminas y actas resulta difícil, y su existencia oscura y por lo tanto su contenido también queda siendo poco conocido.

Los resultados finales del proceso se entregan por carta a los vocales seleccionados y también pueden ser vistos en la página de Consulta de Datos Electorales del SERVEL [10], como se muestra¹ en la figura 2.2.

Con respecto a datos sobre la junta electoral relevante, el vínculo en la fila “Junta Electoral” apunta a “<https://www.servel.cl/juntas-electorales/>” donde se debe buscar la junta electoral relevante dentro de la lista de todas las existentes, y donde no se explica qué son las juntas electorales.

¹Para forzar a que se muestren los datos, se cambiaron estilos que estaban como “`display:none;`” de las últimas dos filas de la primera tabla (rellenando con datos de ejemplo, pues estaban vacías) y de la segunda tabla completa, para poder simular lo que se vería en época de elecciones si se saliese elegido como vocal de mesa.


SERVEL Servicio Electoral de Chile

SERVEL INFORMACIÓN ELECTORAL ESTADÍSTICAS PARTIDOS TRÁMITES PREGUNTAS FRECUENTES SALA DE PRENSA

Consulta de Datos Electorales

Ingrese su RUN:

184123991

I'm not a robot  reCAPTCHA Privacy - Terms

Consultar

DATOS ELECTORALES	
RUN	18412399-1
Nombre	PINO CORDOVA FRANCO ANIBAL
Circunscripción Electoral	PLAZA ÑUÑO A
Comuna	ÑUÑO A
Provincia	SANTIAGO
Región	METROPOLITANA DE SANTIAGO
País	CHILE
Mesa	67M
Habilitado para Sufragar	SÍ
Local de votación	LICEO CARMELA SILVA DONOSO
Ubicación	AV. PEDRO DE VALDIVIA N° 4862, ÑUÑO A

DESIGNACIONES	
Junta Electoral	MACUL (EX ÑUÑO A) - Revisa los datos de contacto aquí
Vocal de Mesa	SÍ.
Miembro Colegio Escrutador	Si cumpliste la función de miembro de Colegio Escrutador el 20 de noviembre, deberás asistir el lunes 18 de diciembre a ejercer el mismo rol.

Figura 2.2: Captura de pantalla de la página de Consulta de Datos Electorales del SERVEL en <https://consulta.servel.cl>, obtenida en Junio del 2019. Se fuerza a que muestre información como si se fuese vocal de mesa para.

2.3. Aleatoriedad Verificable

2.3.1. Fuentes de Aleatoriedad

Existen servicios públicos que emiten datos aleatorios cada cierto tiempo, conocidos como “Randomness Beacons”. En general, lo que estos servicios buscan es que los datos que emiten

sean impredecibles, sin sesgos y confiables. Para esto último, los valores aleatorios pueden ser emitidos por una institución ampliamente considerada confiable, o los valores pueden ser tales que sobre ellos pueden hacerse procesos de verificación, con los que se confirme si fueron (o no) realmente generados de una forma que pueda considerarse como “aleatoria”, es decir, sin posibilidad práctica de haber sido predichos por humanos antes de su generación, ni tampoco de haber sido influenciados o atacados luego de ésta.

El Instituto Nacional de Estándares y Tecnología de los Estados Unidos (NIST) tiene un proyecto de creación y promoción de Randomness Beacons [11]. Promovido por este proyecto, desde Brazil se implementa el “Inmetro’s Randomness Beacon” [12], que basa sus fuentes de entropía en procesos cuánticos, consiguiendo así aleatoriedad verdadera. La aleatoriedad es publicada abiertamente por medio de su API, y puede ser considerada como completamente aleatoria e impredecible, asumiendo que se confía en la entidad detrás de este Beacon. Hay también una coalición de distintas entidades emisoras de aleatoriedad pública, conocida como “League of Entropy”, que se han unido para implementar un sistema de aleatoriedad pública distribuido al que llaman “drand” [13].

Formando parte tanto del proyecto de NIST como del grupo League of Entropy, en Chile existe una iniciativa que implementa y mantiene un Randomness Beacon, desarrollada como parte del proyecto “Random UChile” del Laboratorio de Criptografía Aplicada y Seguridad Computacional de la Universidad de Chile (CLCERT). Emite aleatoriedad pública y verificable, construida usando conceptos criptográficos, matemáticos y prácticos, a fin de que la confianza en la aleatoriedad de los valores dependa lo menos posible de una suposición de confianza hacia el CLCERT como institución. El CLCERT llama a su servicio “Faro de Aleatoriedad” [2].

2.3.2. Faro de Aleatoriedad Verificable del CLCERT

Al inicio de cada minuto, el faro genera 512 bits de aleatoriedad, abiertos para uso público, e impredecibles aunque verificables. Esta aleatoriedad se hace disponible en forma de un conjunto de datos llamado “Pulso de Aleatoriedad”, que puede ser obtenido desde su API en forma de JSON [14]. Dentro de los datos del JSON obtenido, aquel que representa el valor aleatorio del pulso es la propiedad “outputValue”. El CLCERT encadena los pulsos y mantiene registro de ellos (esto siendo parte del diseño que los hace verificables), por lo que se puede volver a acceder a los datos de un pulso que se haya generado con anterioridad.

Para recolectar información aleatoria, el CLCERT usa actualmente como base 4 distintas fuentes externas de entropía:

- Centro Sismológico Nacional: A través del sitio <http://sismologia.cl/links/portada/index.html>, se obtiene cada minuto la información de los últimos sismos, de magnitud 3.0 o mayor, que ocurren en el territorio chileno.
- Radio de la Universidad de Chile: Cada minuto, se obtienen los datos del audio producido por la señal pública de la Radio Universidad de Chile (<http://radio.uchile.cl/>). Actualmente los datos consisten en 20 bloques de 4096 bytes cada uno.
- “Tweets” en Twitter: Cada minuto, se obtienen los 10 primeros “temas del momento” (o

“trending topics”) de Twitter, de entre aquellos que Twitter marca como pertenecientes a Chile. Luego de eso, se hace una búsqueda usando esos temas del momento como palabras clave y se obtienen los “tweets” (nombre de los mensajes públicos en esa plataforma) producidos por usuarios en Chile, que usen esas palabras, entre las marcas de 15 y 30 segundos del minuto.

- Blockchain de Ethereum: Cada minuto, se obtiene el hash de 256 bits del valor del último bloque publicado en la blockchain de Ethereum.

Complementando a las fuentes externas, cada minuto también se genera entropía localmente, con un hardware especializado que, a través del aprovechamiento de procesos cuánticos, genera 512 bits de aleatoriedad. Estos se generan para usar en el pulso del minuto siguiente, haciendo uso de un esquema de compromiso criptográfico (o un “commitment”), que permite ocultar el valor para su uso posterior, asegurando que el valor que se revela al final es el mismo que se mantenía oculto al principio. Finalmente, toda la entropía recolectada se combina a través de un proceso de hash, con lo que se genera el *output* de 512 bits que se convierte en el valor emitido por el faro [2, Sección 2.1. Algoritmo de Generación].

El CLCERT también ha hecho públicos scripts para verificar la correctitud de los pulsos de aleatoriedad, junto a una explicación de cómo usarlos [15].



Figura 2.3: Muestra de los certificados de pulsos de aleatoriedad que ofrece el CLCERT.

Además de esto, se generan certificados con los que se puede avalar que un pulso de aleato-

riedad, con cierto número identificador y con cierto valor aleatorio asociado, fue efectivamente generado. Un ejemplo de esto puede verse en la figura 2.3 (PDF del certificado puede encontrarse en la siguiente dirección: http://web.archive.org/web/20190627080918/https://random.uchile.cl/beacon/2.0/pulse_certificate/chain/4/pulse/295668).

2.3.3. Uso Actual del Faro de Aleatoriedad

En general, los datos que se obtienen de estos servicios serán ocupados para obtener semillas para generadores pseudoaleatorios de números, a partir de los cuales se pueden generar más números que se pueden usar más directamente para implementar funciones relacionadas al uso de azar. Es importante también notar que distintos generadores tienen distintas propiedades, que hacen que unos sean considerados más seguros que otros, en el sentido que los algoritmos usados para algunos generadores pueden tener sesgos que hacen que, sin conocimiento de la semilla, su *output* pueda ser predicho o analizado estadísticamente para encontrar sesgos (más discusión sobre esto y cómo se abordó esta problemática en la sección 2.4).

Un uso notable del faro de aleatoriedad es el trabajo que se está haciendo con la Contraloría General de la República de Chile.

En colaboración con el CLCERT, Contraloría está evaluando e implementando un sistema por medio del cual se conectaría al faro de aleatoriedad. Con motivo de evitar que pueda creerse que hay persecución política contra quien sea auditado, Contraloría usaría el faro para seleccionar a los candidatos a auditar. Se sientan así de cierta manera las bases y una referencia para que otros organismos estatales vean el valor en el concepto detrás del faro.

En contraste con la implementación para el sorteo de vocales de mesa que se busca implementar, en el caso de Contraloría la obtención y uso de aleatoriedad están centralizados a una sola entidad (Contraloría misma). En el caso del sorteo de vocales de mesa, el software usado para registrarlo es provisto por el SERVEL, mientras que las distintas juntas electorales del país son quienes lo usan.

La parte de verificación de lo hecho con Contraloría está más bien pensada para ser hecha por organizaciones que puedan acceder a algún grado de conocimiento técnico, como ONGs y Contraloría misma (es decir, que ésta pueda verificarse a sí misma el correcto uso del faro). Mientras que en esta memoria se apunta a habilitar, en todo lo posible y razonable, a cualquier persona que haya sido sorteada como vocal de mesa a verificar su propia selección.

2.4. Consideraciones sobre Cómo Elegir Números al Azar

Si bien aún no se conocen exactamente las condiciones, ambientes y herramientas que se tendrían a disposición al desarrollar para el SIVOME directamente en conjunto con el SERVEL, más allá de que es una aplicación web (y todo lo que esto implica), hay asuntos y sutilezas importantes con los que generalmente se debe tener cuidado a la hora de implementar

formas de elegir números al azar.

Como punto de partida, está la pregunta de si usar un valor aleatorio, como lo puede ser el de un pulso de aleatoriedad, directamente, o si es mejor dar este valor como semilla a un generador pseudoaleatorio.

Usar el valor directamente supone tener una cantidad limitada de bits aleatorios a disposición de la cual derivar números concretos, mientras que usarla como semilla para un generador permite obtener una cantidad mucho más grande de números de ella. En el peor de los casos, si se usan todos los bits del valor aleatorio se tendría que esperar hasta poder conseguir otro para poder continuar con la generación de números, lo que puede ser un problema notable si el tiempo de espera es de un minuto. Se considera que es mejor encontrar un buen generador pseudoaleatorio de números, a modo de evitar en todo caso práctico la posibilidad de tener que necesitar más aleatoriedad. Esto aporta también cierta flexibilidad ante la eventualidad de tener que usar más aleatoriedad o generar más números de lo que, en un principio, se cree es necesario.

Tomada esta decisión, la primera sutileza es fijarse en qué sucede con la semilla dada al generador de números aleatorios.

En principio, al realizar la elección se debería elegir solo desde $\frac{30!}{(30-10)!}$ alternativas posibles (el número de formas de elegir 10 elementos de 30, donde el orden sí importa), lo que da un número de elecciones que, por ejemplo, no se puede describir con solo 32 bits.

Los 512 bits dados por un pulso del faro CLCERT son suficientes para realizar lo necesitado, pero sería bueno poder utilizar implementaciones ya existentes de generadores pseudoaleatorios. Si se da la semilla de 512 bits, sería esperable que el generador funcione como se requiere, y esto en algunos casos, pero no siempre, es así. Como ejemplo, en la librería `random` de Python 2 [16], las semillas pasan por un proceso de hashing si es que no son `int` o `long`, que las convierte en semillas de 32 o 64 bits, dependiendo de la plataforma. Así, no se pueden generar todas las permutaciones necesitadas en el caso de 32 bits, y el proceso estará sesgado a ciertos órdenes [17]. Es importante entonces saber qué sucede con la semilla en los generadores que se quieran usar.

Una segunda sutileza en la que hay que fijarse puede ser la implementación de métodos para escoger enteros al azar entre dos límites, usualmente de 0 a algún número m (por simplicidad de lo que sigue, m exclusivo), la cual es una función que se usaría dentro de la función para elegir 10 elementos de entre 30. Si el número m no es una potencia de 2, y dado que las semillas suelen procesarse como potencias de 2, en principio podría haber números favorecidos. A modo de ejemplo: sea $m = 30$, entonces se usarán 5 bits de aleatoriedad para realizar la elección, esto es, hay 32 configuraciones que el método de selección puede usar. 30 de ellas cubren los 30 números posibles deseados, pero entonces queda el problema: ¿qué hacer con las 2 configuraciones sobrantes?

Una implementación problemática sería aplicar módulo del número de interés, de modo que las configuraciones 31 y 32 resulten en elegir 0 y 1, pues esto implica que hay un sesgo hacia esos números, ya que las primeras dos configuraciones también los eligen. La solución debería ser que si se llega a las configuraciones 31 y 32 entonces se vuelve a intentar generar

una nueva hasta llegar a una correcta. Lo explicado aquí es una simplificación, pero debe notarse que sí hay implementaciones reales conocidas por tener sesgos, verbigracia `rand` en C, motivo por el cual la documentación de C++ recomienda la función `random` en su lugar [18].

Varias plataformas y librerías ofrecen maneras de elegir elementos al azar de una lista, pero lo hablado aquí muestra que puede ser importante fijarse e investigar en cómo exactamente.

En cualquier caso, sin importar qué métodos soporte nativamente el lenguaje en el cual se deba trabajar, es una buena apuesta buscar librerías externas o, de no existir, se pueden implementar manualmente métodos de elección al azar con las características deseadas, por ejemplo, si no se tiene un método confiable para elegir muestras de tamaño k al azar a partir de una lista, no es complicado implementar una variante del algoritmo Fisher-Yates [17, Descripción del Algoritmo P] que se detenga luego de reordenar k elementos (en lugar de dejarlo completar el reordenamiento de toda la lista), que es una opción atractiva en cuanto a que no necesita cambiar el tamaño de la lista en la que trabaja. Por otro lado, también se puede simplemente obtener un índice al azar de la lista, obtener el elemento en ese índice, eliminarlo de la lista y repetir el proceso cuanto sea necesario, que tiene la ventaja de ser un método bastante intuitivo, lo que es deseable para poder explicar el algoritmo a la ciudadanía.

Una implementación de generador de números pseudoaleatorios que se ha considerado, para la cual existen librerías de terceros para varios lenguajes, es “Fortuna” [19]. En general, implementaciones de Fortuna usan suficientes bits en sus semillas como para cubrir aplicaciones relevantes para esta memoria, y se entiende como un método criptográficamente seguro, lo que da garantías que cubren lo buscado. Por otro lado, también hace garantías más fuertes de lo necesario para esta memoria (por ejemplo, no interesa evitar la deducción de números ya generados), y en parte para poder hacer esas garantías que éste método se puede considerar complejo en comparación a otras posibles soluciones.

Dado que el SIVOME es una aplicación web, y dado que se espera que el sistema de verificación pueda ser verificado por internet, se investigan librerías y formas de implementar la selección en JavaScript. Lo primero en investigarse fueron implementaciones de Fortuna. Sin embargo, lo que se quiere lograr es una selección reproducible, y en general las implementaciones encontradas tienen por objetivo montar un sistema que recolecte entropía constantemente, que es parte de lo necesario para que Fortuna cumpla los objetivos que tiene, pero que también se traduce en que hacer elecciones reproducibles se vuelve algo más complejo. Si bien se puede, lograr lo que esta memoria se propone implica limitar el funcionamiento de cualquier implementación de Fortuna y de cierto modo negar algunas de las ideas principales detrás de sus algoritmos, esto se refleja en que varias implementaciones con las que se experimentó no tienen forma de lograr reproducibilidad por defecto.

Debido a esto, en su lugar, se buscan otros generadores pseudoaleatorios considerados seguros, y así se llega a “ChaCha20” [20], un cifrador de flujo con el que se puede obtener un generador pseudoaleatorio de números que también es ampliamente considerado seguro (a modo de ejemplo: kernels recientes de Linux usan implementaciones basadas en ChaCha20 para `/dev/urandom`) [21].

En la sección 4.3 se describe más específicamente cómo se implementa y usa la generación de números pseudoaleatorio para el trabajo de esta memoria.

Capítulo 3

Descripción del Problema

3.1. Problema y Oportunidades

Como se describe en la sección 1.1 (Introducción: Contexto), el proceso del sorteo por medio del cual se designan vocales de mesa ocurre de tal manera que su verificabilidad tiene limitaciones. Una de ellas es la necesidad de obtener documentos de forma física, otra es tener que presenciar el evento en el momento en que se lleva a cabo. Además de esto, hay una falta de conocimiento por parte de la ciudadanía chilena acerca de cómo funciona la designación de vocales.

Si bien los procedimientos sí están registrados y sí se pueden fiscalizar de forma presencial, en la práctica los detalles del proceso tienden a ser poco conocidos, y la información asociada a sus resultados no es fácilmente accesible a toda persona directamente afectada.

Para ir aún más lejos, la existencia de aleatoriedad verificable da cuenta de que se puede aumentar la transparencia y verificabilidad de la selección de vocales de mesa.

Hay también una oportunidad en la existencia de la página de Consulta de Datos Electorales del SERVEL, en cuanto a que es muy común ocupar dicho sitio para averiguar si se debe o no cumplir con el rol de vocal de mesa. Teniendo que ya se busca colaborar con el SERVEL para la implementación de aleatoriedad verificable, se observa la posibilidad de modificar esa página para ofrecer más información y recursos sobre los procedimientos relacionados a la selección de vocales de mesa y a las juntas electorales, a fin de evitar problemas que puedan darse por falta de conocimiento sobre esas áreas.

3.2. Relevancia

Existe una desconfianza y escepticismo en algunas personas sobre su designación como vocales de mesa. Vale la pena tomar en cuenta y abordar esa desconfianza, pues los vocales de mesa son parte importante del sistema político chileno, por ejemplo porque tienen acceso

directo a las urnas con los votos de los procesos electorarios del país, y se encargan de su primer escrutinio. Es por ello que se considera muy positivo lograr que el proceso sea más transparente y fácilmente verificable

Llevar esta herramienta al público (en específico: el concepto de verificar cómo se les eligió como vocales de mesa) implica que se le debe dar información a la ciudadanía sobre qué es y cómo se usa, lo que conlleva explicar el proceso de selección de vocales mismo. Así, un efecto de implementar las ideas detrás de este trabajo de título es que se crea una fuente de información más para las personas sobre los procesos democráticos. Al considerar, por ejemplo, que el SERVEL recibe y debe lidiar con quejas relacionadas a la selección de vocales, cuando en realidad esta clase de reclamos normalmente debiesen estar dirigidos a la junta electoral asociada a la circunscripción de quien haga el reclamo, se pone en evidencia un vacío en el conocimiento general de la ciudadanía. Se puede apreciar así el beneficio de tener nuevas fuentes de información.

En un sentido más amplio, la existencia de organizaciones como Vota Inteligente [22], y Chile Transparente [23], son una muestra de iniciativa y deseo por parte de la ciudadanía chilena de avanzar en temas de transparencia y disponibilidad de información sobre procesos democráticos. A nivel internacional, que principios como “el derecho a explicación” (“*Right to Explanation*”) existan y sean perseguidos, demuestra también este deseo de transparencia sobre las decisiones que los gobiernos toman sobre las personas bajo sus jurisdicciones [24]. Este derecho es más específicamente discutido en términos de los resultados de procesos computacionales, así, defender el derecho a explicación (considerado en esta memoria como una intención positiva) implica también que se dé información sobre el sorteo verificable a la ciudadanía chilena.

En una línea similar, se considera deseable traer la clase de sistemas aquí propuestos al ojo público, y darle atención a la aleatoriedad verificable como una posible herramienta más que puede ayudar a transparentar asuntos importantes.

La aleatoriedad verificable, no es necesariamente un concepto nuevo, pero el público general, o en efecto incluso gran parte de aquellos con conocimiento técnico y/o formal sobre computación, no tiene demasiado conocimiento de ello, pues incluso en el mundo de la ciencia de la computación es aún un asunto bastante de nicho. Por lo tanto, en la actualidad lo natural es que no se espere o demande el uso de aleatoriedad verificable de organismos que, en realidad, sí podrían usar estos sistemas para dar mejores garantías de la correcta ejecución de sus tareas. Se propone entonces que implementando estos sistemas se puede aportar, en alguna medida, a acostumbrar a la ciudadanía a esperar transparencia y disponibilidad de los procesos relacionados al funcionamiento de la democracia del país. En este sentido, avanzar en la implementación de aleatoriedad verificable en la selección de vocales de mesa puede ayudar a difundir y validar el concepto, potenciando la posibilidad de que se implemente la aleatoriedad verificable en otros sistemas donde pueda ser útil o incluso importante hacerlo.

3.3. Especificaciones de la Solución

3.3.1. Características Generales

En primer lugar, para lo hecho para esta memoria se debe reparar en que no se está trabajando aún directamente con el SERVEL. Por lo tanto, el trabajo se enfoca en crear prototipos y en la confección y demostración de diseños potencialmente aplicables al ambiente real. Se espera que esto ayude en la búsqueda de formalizar un trabajo en conjunto con el SERVEL y las juntas electorales.

La solución debe buscar crear prototipos de un sistema para realizar el sorteo de vocales de mesa usando aleatoriedad verificable, para el uso de juntas electorales, y de un sistema de verificación de estos sorteos.

La creación de estos a su vez necesita otras partes importantes, como lo son implementar una manera de elegir números al azar para usar en el sorteo, y elegir una fuente de aleatoriedad verificable y diseñar cómo usarla para los propósitos de los sistemas de sorteo y verificación.

En términos generales, se entiende que la solución consiste en:

- Implementar un generador de números pseudoaleatorios, que pueda ser usado tanto en el sorteo como en el proceso de verificación del mismo.
- Elegir una fuente de aleatoriedad verificable.
- Crear un prototipo para el sorteo con aleatoriedad verificable.
- Crear un prototipo para la verificación de dicho sorteo.

Se discute cada punto en más detalle a continuación.

3.3.2. Generación de Números Pseudoaleatorios

Con el objetivo de realizar la selección de vocales al azar, es decir, la selección de 10 números sin repetición del 1 al 30, se identifican las siguientes propiedades con las que es importante que el algoritmo de generación de números pseudoaleatorios cumpla lo siguiente:

- Debe ser un generador considerado seguro.
- La selección de números debe ser reproducible.
- Idealmente, no debe ser complejo traducir la implementación de un lenguaje de programación a otro.

El hecho mismo de ser considerado seguro implica generalmente que hay conocimiento y escrutinio por parte de la comunidad criptográfica, lo que en sí mismo implica que el algoritmo sería conocido y se espera entonces encontrar implementaciones de terceras partes que se puedan usar. Al mismo tiempo, ser seguro implica que no hay ataques conocidos para encontrar sesgos, lo que es imperativo de tener.

Es deseable también que no dependa demasiado de las funciones o librerías de un solo lenguaje, pues se quiere que tanto el frontend como el backend del sistema de selección puedan usar el generador (esto para asegurar consistencia, y porque en principio no es tan difícil manipular lo que el frontend en JavaScript pueda enviar al backend). Además sería ideal si terceras partes que quieran hacer la verificación del proceso no encuentren dificultades innecesarias, como dependencia fuerte en funciones o librerías que solo se encuentran fácilmente en uno o pocos lenguajes en particular.

3.3.3. Elección de Fuente de Aleatoriedad Verificable

Una fuente de aleatoriedad verificable en principio debe ser tal que pueda verificarse la correctitud de la aleatoriedad generada, y donde no haya motivo para creer que puede haber manipulación de los datos, o, mejor: que esté implementada a modo que una manipulación sea impracticable. Para la verificación, se busca que la fuente defina propiedades que describan su integridad y puedan ser corroboradas.

Otro punto es que sea una fuente pública, pues la verificación debe poder ser potencialmente realizada por cualquier persona. Se estima también que el hecho de que los datos estén abiertos al público (en vez de, por ejemplo, que solo sean accesibles a través del SERVEL) es saludable para aminorar posibles sentimientos de escepticismo y desconfianza.

Fuera de esto, se buscan herramientas que sean convenientes. Por ejemplo, ayuda considerablemente si hay scripts públicos de verificación implementados, si existen explicaciones públicas de cómo funcionan que puedan ser luego dadas a las terceras partes verificadoras, y si existe una comunidad o grupo de personas que se pueda contactar para resolver dudas que puedan surgir durante el trabajo de memoria.

3.3.4. Sistema de Sorteo

Con el prototipo de sistema de sorteo se debe:

- Permitir elegir 10 números al azar en secuencia y sin repetición, dentro del rango de números del 1 al 30 (inclusivos).
- No permitir que haya forma práctica de realizar el proceso normal de confección de la nómina de candidatos a vocales de mesa con conocimiento, o estimaciones, del resultado del sorteo.
- Identificar los datos necesarios que se deben tener en la base de datos para el sorteo verificable.
- Implementar que dichos datos sean registrados y enviados a la base de datos.
- Implementarse a conciencia de evitar reticencia al cambio por parte de las juntas electorales.

Para elaborar más: el sorteo mismo, se basará en el generador pseudoaleatorio de números que se implemente, y el algoritmo resultante no debe introducir sesgos. Pero se debe considerar

además, que no importa que tan perfectamente aleatorio sea el sorteo, si las nóminas pueden confeccionarse con conocimiento, por ejemplo, del pulso de aleatoriedad que se usará, entonces los resultados siguen siendo potencialmente manipulables. Esto podría ser por medio de elegir candidatos de tal forma que una persona específica quede indexada con un número que se sabe va a ser elegido.

El SIVOME está diseñado para que las nóminas deban estar listas antes del sorteo, lo que ayuda a que las juntas no puedan hacer el sorteo y luego elegir candidatos ya sabiendo si serán elegidos o no. Sin embargo, con la introducción de fuentes de aleatoriedad, se introduce también la posibilidad de conocer el pulso de aleatoriedad (y por ende los resultados finales) antes de realizar el sorteo formal, por lo que se considera necesario implementar el sistema nuevo con esto en mente.

Debe también diseñarse cómo se guarda la información necesaria para el sorteo verificable, a modo de poder identificar los cambios que tendrían que hacerse no solo en las interfaces, sino que también en el backend del ambiente real para poder implementarlo.

El sistema de sorteo debe hacerse orientado al uso por la juntas electorales, entendiendo que hay muchas juntas electorales y que cada una actúa independientemente (dentro de sus deberes). En lo posible las interfaces no deben presentar un cambio brusco a lo que ellas ya acostumbran ocupar, con este fin también es bueno considerar ofrecer esto siempre como una simple opción a lo actual, y no un completo reemplazo.

3.3.5. Sistema de Verificación

Para el sistema de verificación, se busca que con el prototipo se demuestre cómo:

- Guardar la información del sorteo para permitir la posterior verificación del mismo.
- Comunicar no solo los resultados del sorteo (como es el caso actual) si no que también detalles del procedimiento que llevó a ellos, por lo menos para el caso en el que se use el sorteo verificable.

Elaborando en estos puntos: diseñar o identificar cambios a la base de datos para que guarde la información del sorteo verificable, como se describe en la sección anterior que debe hacerse, implica también que se registren los datos necesarios para poder realizar la verificación.

La verificación en este caso no se da solo por poder verificar la integridad del pulso de aleatoriedad, sino que implica también poder recrear la forma en la que se usó, a modo de poder confirmar los resultados finales del sorteo. Así, se debe tomar en cuenta, no solo el registro de los resultados del sorteo para el uso del SERVEL y las juntas electorales, sino que también que se quiere que el procedimiento para llegar a ellos pueda ser verificado por terceras partes.

En ese sentido, por lo menos, deben poder hacerse de conocimiento público: el algoritmo o estándar usados para ordenar la nómina de candidatos a vocales de mesa, el generador

pseudoaleatorio de números, la semilla dada al generador, el pulso de aleatoriedad asociado a dicha semilla, y el algoritmo que realiza el sorteo. Como mínimo, una persona con conocimiento técnico debiese poder usar los recursos públicos para recrear el sorteo y verificar el pulso de aleatoriedad, según los procedimientos de verificación que defina la fuente del mismo.

En una línea similar, se considera también muy deseable poder comunicar conceptos relacionados al sorteo verificable a vocales de mesa que se elijan de esa manera. Esto pues se considera que parte importante de que haya un sorteo *verificable* es, en lo posible, habilitar la efectiva verificación del mismo, lo cual incluye crear conciencia del concepto y dar recursos que permitan a cualquier persona ganar un sentido de que puede demostrarse la correctitud del procedimiento.

Capítulo 4

Solución

4.1. Diseño General

En este capítulo se describe cómo se diseñan las interacciones entre los distintos sistemas involucrados, así como las intervenciones que se busca hacer en cada uno de ellos.

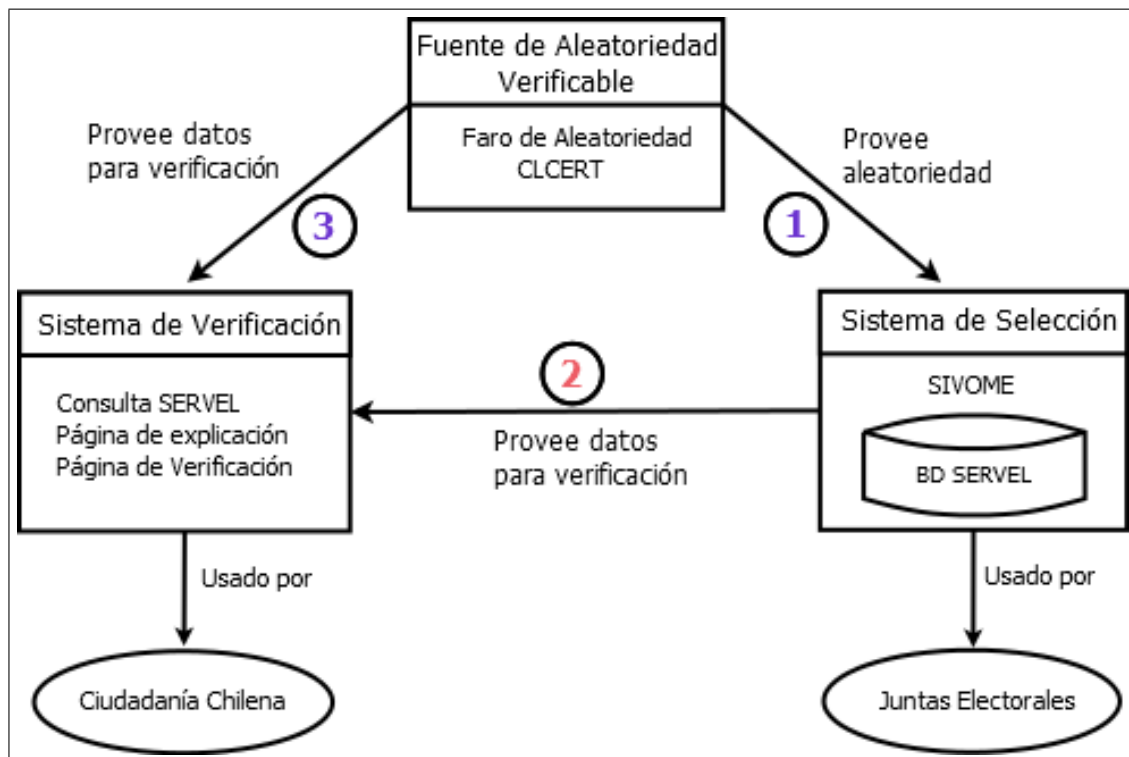


Figura 4.1: Diagrama general de los distintos sistemas relevantes para la implementación de aleatoriedad verificable en las elecciones de vocales de mesa. Interacciones numeradas para facilitar referencia.

A grandes rasgos, se entiende que existen tres sistemas con los que se debe trabajar:

1. Fuente de Aleatoriedad Verificable.
2. Sistema de Selección de Vocales de Mesa.
3. Sistema de Verificación.

En la figura 4.1 se puede ver un diagrama general de las partes y sus relaciones. Las siguientes secciones de este capítulo explican en más detalle los papeles de cada sistema, las interacciones entre ellos, y cómo se implementan. Primero se describen los puntos que funcionan como fundamentos para la solución: el procedimiento para sortear, la manera en la que se generan números pseudoaleatorios y el backend.

Para la descripción de los últimos dos puntos en la lista, se empieza con una sección donde primero se hablará de los planes que hubo para intervenir los sistemas existentes de selección o verificación. Luego se detalla y presenta el trabajo hecho que sigue esos planes.

En general, para las interfaces que se muestran en las siguientes secciones se usó el framework JavaScript “Vue.js”, con el que se pueden hacer aplicaciones e interfaces web reactivas. Para realizar peticiones, se usó la librería para JavaScript llamada “Axios”. Tanto las generalidades como los detalles del lado del servidor se pueden encontrar en la sección 4.5.

4.2. Procedimiento del Sorteo

4.2.1. Preámbulo

Independiente de qué herramientas se usen (lenguaje de programación, base de datos, librerías, etc...), se define un procedimiento que se sigue para sortear, el cual tiene que no solo implementarse para el uso de las juntas electorales, sino que también es el que terceras partes verificadores tienen que replicar para poder confirmar los resultados.

A grandes rasgos, se siguen tres pasos: obtención de aleatoriedad, procesamiento de aleatoriedad (es decir, su conversión en semilla para generación de números pseudoaleatorios), y el sorteo mismo de vocales de mesa.

4.2.2. Obtención de Aleatoriedad

Con respecto a obtener la aleatoriedad, se usará una fuente de aleatoriedad verificable de tipo Randomness Beacon, más específicamente, el servicio del faro de aleatoriedad del CLCERT. Se puede así conseguir un pulso de aleatoriedad en forma de JSON a través de la API del faro. Los 512 bits de aleatoriedad del pulso están dados como un string de 128 caracteres en formato hexadecimal. Se discute más el rol de la fuente de aleatoriedad para con los sistemas implementados en la sección 4.4.

4.2.3. Procesamiento de la Aleatoriedad

Para usar este valor aleatorio para la generación de números, se define un generador de números pseudoaleatorios (más detalle en la sección 4.3), el cual recibe una semilla de 320 bits.

Se sigue también un procedimiento para individualizar la semilla a cada junta electoral, a fin de evitar que todas las juntas electorales obtengan los mismos resultados en sus sorteos (lo que podría ser percibido como algo extraño si ocurriese). Para esto, se toma el valor aleatorio del pulso, el nombre de la junta electoral y se le dan a la función de hash SHA-3 512. Los primeros 320 bits del hash producido son finalmente usados como semilla para el generador. Se usa SHA-3 512 ya que es ampliamente considerada como una función de hash segura [25], por lo que usarla no introduce ninguna vulnerabilidad o sesgo que puedan ser prácticamente aprovechados.

La combinación del valor aleatorio y el nombre de la junta electoral se crea convirtiendo a ambos en secuencias de bytes. Para el valor aleatorio, simplemente se pasa de su representación en hexadecimal a su representación en bytes, luego a esta se le concatena la secuencia de bytes obtenida de codificar el nombre de la junta según UTF-8.

A partir del generador, se implementa un método para obtener números enteros en un rango $[0, n]$, para n entero positivo, el cual se usa dentro del algoritmo de sorteo. La forma en la que se llega a esta funcionalidad está descrita en la sección 4.3.2.

4.2.4. Algoritmo de Sorteo

El algoritmo de sorteo se basa en simplemente escoger un elemento al azar de una lista, guardarlo y eliminarlo de la lista. Repitiendo esto cuanto sea necesario para obtener la cantidad deseada de elementos escogidos al azar.

Más específicamente. Si se tiene una lista L de largo N , y una lista donde se guardan los elementos seleccionados S , se puede definir una iteración del algoritmo como:

1. Sea N el largo actual de L .
2. Sea k un número al azar en $[0, N - 1]$
3. Insertar $L[k]$ (indexando desde 0) al final de S .
4. Remover $L[k]$ de L .

Es fácil ver que el algoritmo mismo no introduce sesgos. Además de ello, se elige este algoritmo para realizar el sorteo pues es simple y muy similar conceptualmente a cómo se realizaría, por ejemplo, con una tómbola. La simplicidad se estima importante aquí, pues se considera importante poder explicar, en lo posible, cómo se hace el sorteo a la ciudadanía. Otros algoritmos vistos para la selección de números aleatorios desde una lista, sin repetición y en secuencia, son bastante menos intuitivos sin necesariamente dar ganancias significativas en otros aspectos.

Así más en concreto, para obtener 10 números al azar del 1 al 30, sin repetición, se construye una lista de los números del 1 al 30, ordenados de menor a mayor, y se repiten los pasos arriba definidos 10 veces sobre ella.

Por otro lado, se define también un sistema de “reintentos”. Esto significa desechar un sorteo y proseguir a intentar sortear de nuevo. Se hace esto con el fin de dar más flexibilidad a las juntas electorales, como una forma de suavizar una posible percepción de imposición o falta de control, que pueda surgir al pasar a usar el sorteo digitalmente verificable.

Para el algoritmo, reiniciar significa (usando los nombres antes dados para las listas) que se reinicia la lista *L* a su estado original (un rango ordenado de números del 1 al 30) y se vacía la lista *S*. Dado que el generador no se inicializa de nuevo, este dará (con extremadamente alta probabilidad) resultados distintos a los del intento anterior. Cualquier reintento realizado debe ser registrado, para luego formar parte de la información publicada sobre el sorteo verificable, pues de otro modo no se pueden verificar los resultados.

Para aplicar los 10 números sorteados a una mesa receptora en particular, se toma su nómina de candidatos a vocales de mesa y se ordena alfabéticamente según el orden dado al español de acuerdo al Unicode Collation Algorithm¹. Con la nómina ordenada, se pueden entonces identificar a las personas que salen sorteadas al hacer una correspondencia entre los números dados por el algoritmo de sorteo y su posición en la lista. El orden en que las personas son sorteadas es dado por el orden en que el algoritmo escoge los números (de forma que se siga la letra de la ley, que implica que los números son sorteado en secuencia), de forma que así se obtiene su prioridad en la lista final de vocales titulares y reemplazantes, siendo las de mayor prioridad (las primeras personas en ser sorteadas) las que son primero llamadas a cumplir su rol de vocales de mesa, cuando sea necesario.

En todo esto simplemente se toma el supuesto de que existe un generador de números al azar el cual pueda usarse dentro del algoritmo. En la siguiente sección se habla en más detalle cómo éste se implementa.

4.3. Librería de Funciones Aleatorias

4.3.1. Descripción General

Teniendo en mente lo descrito en la sección 3.3.2, ChaCha20 es una buena fundación en la cual basarse para la generación de números pseudoaleatorios. Es un cifrador bastante conocido, no es inusual encontrarlo en librerías criptográficas, como por ejemplo: se encontraron varias implementaciones en JavaScript y está también en PyCryptodome [26], una librería criptográfica para Python. Además, ChaCha20 es bastante razonable de implementar directamente de ser necesario.

Para el trabajo de memoria se usa “chacha20-js” [27], como base para la implementación

¹Para ser aún más específicos, se usa el orden dado por la librería PyICU, usando un objeto `icu.collator` (que tiene la funcionalidad de ordenar) cuya propiedad `locale` es la identificada por el código ‘es’.

de un generador pseudoaleatorio de números en JavaScript. Se le eligió pues, a diferencia de las alternativas encontradas, es más flexible al no estar implementada específicamente con una interfaz de cifrador, sino que directamente entrega el bloque de 64 bytes pseudoaleatorios que se genera en cada llamada al algoritmo ChaCha20 (que es lo subyacente a los cifradores ChaCha20 que simplemente obtienen ese bloque de 64 bytes y luego aplican la operación XOR para encriptar mensajes).

Con esa implementación como base, se construye “ChaCha20-Generator-Utilities” (GitHub: <https://github.com/FrancoPinoC/ChaCha20-Generator-Utilities/tree/master/js>), una librería para JavaScript con varias funciones relacionadas a la generación de números pseudoaleatorios.

La implementación de ChaCha20 de `chacha20-js` usa una clave de 32 bytes y un IV de 8 bytes, por esto la librería aquí implementada usa una semilla de 320 bits, que internamente se usa para obtener la clave y el IV. En la librería se define un objeto `ChaChaGen`, a través de cuyos métodos se accede a las funcionalidades ofrecidas por la librería. Este objeto debe ser inicializado con un string de 80 caracteres en formato hexadecimal (en este formato para que sea fácil usar los valores que entregan los pulsos del CLCERT directamente), el cual representa a la semilla de 320 bits antes mencionada.

En esta librería se implementan varias funciones comunes de librerías de números pseudoaleatorios, como lo son obtener un cierto número de bytes aleatorios, obtener un `float` en el rango $[0, 1)$, o un número entero dentro de un rango.

Además de funciones para obtener distintos formatos de números aleatorios, se implementan funciones que utilizan la aleatoriedad para distintos fines, como elegir uno o más números al azar dentro de una lista, obtener muestras aleatorias y reordenar secuencias.

Las últimas dos funciones mencionadas se implementaron usando una modificación del algoritmo Fisher-Yates de base, en la que se parametriza el número de pasos que da el algoritmo, de modo que se pueda detener antes de completar el reordenamiento. Esto es útil pues en cada paso, el algoritmo selecciona un elemento de la secuencia y lo deja al final de la parte no reordenada de la misma, esto quiere decir que al detenerlo en el paso n y retornar en orden invertido los últimos n elementos, se obtiene una secuencia de n elementos de la lista obtenidos al azar (se extiende del hecho de que Fisher-Yates completo no tiene sesgos, que este algoritmo parcial tampoco los tiene).

Otro punto interesante de la librería es la implementación de una manera de ahorrar entropía para el caso en que se quiera obtener una muestra de tamaño mayor a la mitad de N (tamaño de población a muestrear), donde el orden no importe. En ese caso, primero se obtiene una muestra de tamaño $N-s$ (con s tamaño de la muestra buscada), y luego se “niega” esa muestra, en el sentido de que ésta pasa a representar el conjunto de elementos *no* escogidos (idea inspirada en una entrada del blog de Daniel Lemire [28]). Debido a cómo funciona el algoritmo Fisher-Yates, hacerlo de esta manera permite ahorrar entropía para el caso descrito, pues se necesitan generar menos números aleatorios, en comparación a obtener la muestra de forma directa.

Cabe notar también, que la librería fue rápidamente adaptada a Python por Camilo Gómez

[4] (desarrollador principal de Random UChile), lo que ayuda a demostrar que se cumple la intención de que la librería no fuese demasiado compleja de traducir a otros lenguajes. Esta versión es usada para implementar la verificación del sorteo en el backend de los sistemas implementados, que está hecho en Python.

4.3.2. Generando un Número en un Rango

Para llegar a poder implementar la selección sin repetición de elementos de una lista, se necesitan implementar ciertas funciones como cimientos sobre los cuales construir el algoritmo final de selección, a continuación se explica el camino para llegar a lo buscado.

Como punto de partida, se tiene la implementación de ChaCha20 mencionada anteriormente sobre la que se trabaja, más en específico esto significa que se tiene una función con la cual, dada una semilla, se pueden conseguir bloques de 64 bytes pseudoaleatorios. En la librería, es el método `_newRandBlock()` el que se encarga de llamar a la función ChaCha20 subyacente usada para generar un nuevo bloque, el cual se guardará en forma de arreglo `Uint8Array`, como miembro del objeto `ChaChaGen` para su posterior uso. El método `_newRandBlock()`, como el hecho de que tenga un guión-bajo como prefijo indica, se debe entender como un método privado, diseñado para ser usado solo dentro de otros métodos de la librería.

Ya que se tenga lo descrito en el párrafo anterior como base, lo siguiente es extraer bytes de ese bloque para usarlos. En la librería se implementa `getBytes(n)`, que recibe el número de bytes que se desea obtener. Este método va consumiendo los bytes del bloque y llama a `_newRandBlock()` si necesitase generar uno nuevo para completar la cantidad de bytes pedida. “Consumir los bytes” aquí simplemente significa que se va accediendo a los elementos del bloque, copiándolos a un nuevo arreglo de tipo `Uint8Array` para retornarlos, y avanzando un índice que se mantiene como variable de instancia para saber a qué elementos acceder en la siguiente llamada. Este método cumple el rol de tomar lo dado por `chacha20-js` y transformarlo a un formato más simple de usar, tanto para usuarios como por otros métodos en la librería.

El siguiente paso a seguir, es convertir esos bytes pseudoaleatorios a números propiamente tales. Para conseguir esto primero se implementa el método `getRandBits(nbits)`, que usará los bytes pseudoaleatorios para construir una secuencia de bits pseudoaleatoria, la cual se da en forma de número entero entre 0 y $2^{nbits} - 1$ (inclusivo), tal como la función `getrandbits(k)` del módulo `random` de Python lo hace.

Se implementa luego `randUInt(max)`, que calcula $\lceil \log_2(\max) \rceil + 1$, el número de bits mínimos necesarios para expresar cualquier número entero en el rango $[0, \max]$, y pasa esto como argumento al método `getRandBits`. En este caso `getRandBits` puede retornar un número mayor a lo buscado, en cuyo caso simplemente se vuelve a llamar con el mismo argumento, iterando hasta conseguir un número dentro del rango.

Lo aquí descrito permite obtener un número entero en un rango $[0, n]$ (con n siendo algún entero positivo), a partir de lo cual se puede implementar el algoritmo que se necesita para

llevar a cabo el sorteo.

Cabe mencionar que para la implementación en JavaScript se limita la cantidad de bits aleatorios que pueden pedírsele a `getRandBits` a 53. Esto porque JavaScript define `Number.MAX_SAFE_INTEGER= 253 - 1` como el máximo número entero positivo que se puede representar de forma precisa y segura, por lo que intentar ir más allá de 53 bits puede dar resultados erróneos o ambiguos (si bien existen librerías externas que pueden ayudar a representar números de más bits).

4.4. Fuente de Aleatoriedad Verificable

El servicio de aleatoriedad pública y verificable usado es el Faro de Aleatoriedad del CLCERT. El faro provee una manera fácil de obtener una semilla para el generador pseudoaleatorio cada minuto. El hecho de que la iniciativa provenga de la misma nación que la que enmarca el problema a resolver, y desde una institución bien reconocida como lo es la Universidad de Chile, se espera pueda ayudar a crear confianza con las juntas electorales, el SERVEL y terceros que deseen verificar. Siendo también una gran ventaja que se pueda trabajar y consultar con miembros de la organización directamente. Que el sitio web del proyecto Random UChile esté en español también es una ventaja en términos de dar recursos a la ciudadanía.

Ya que se usa un servicio público ya existente (en lugar de implementarse uno nuevo), lo que es más relevante para discutir en esta sección es cómo se usa dicho servicio; para el diseño de la selección verificable de vocales de mesa, se hace que el faro de aleatoriedad interactúe de las siguientes maneras con los otros sistemas:

- Sistema de Selección: da el pulso de aleatoriedad, cuyos datos se usan para obtener una semilla para los procesos aleatorios necesarios para realizar la selección. Es la interacción número 1 en la figura 4.1, y corresponde a que el sistema de selección haga un llamado a la API del faro aleatorio para obtener un pulso, en principio usando “<https://beacon.clcert.cl/beacon/2.0/pulse/last>”, con lo que se obtiene el JSON que representa al pulso de aleatoriedad.
- Sistema de Verificación: es la interacción número 3 en la figura 4.1 y corresponde a que el faro provea los datos necesarios para realizar la verificación. Dado el identificador del pulso usado en la selección, el sistema de verificación puede correr un programa como el que ha desarrollado el CLCERT [15] para verificar, el cual usará la API del faro para obtener los datos relevantes para la verificación de ese pulso en específico. El valor aleatorio del pulso es también un dato clave para poder recrear el sorteo.

En las siguientes secciones se detallará exactamente como los sistemas interactúan con el faro y como usan los datos que obtienen del él.

4.5. Backend

4.5.1. Construcción del Backend

Para los prototipos hechos se necesita poder persistir datos, así como tener servicios web para obtener información sobre ellos o para modificarlos. Para cubrir estas necesidades, así como con el propósito de identificar qué se necesita para implementar los sistemas de sorteo verificable en un ambiente real, se construye un backend de propósito general, con un servidor de desarrollo y una base de datos.

El backend se programa en Python, usando el framework para aplicaciones web Flask. Se elige Flask pues es simple, flexible, no hace imposición fuerte de estructuras y permite obtener funcionalidad rápidamente. En el sentido en que lo que se hace es un prototipo, todas estas cualidades se consideraron altamente beneficiosas. Además, si bien Flask es bastante simple, existen muchas librerías hechas para extender su funcionalidad, de manera que se pueda evolucionar el proyecto en caso de ser necesario.

La base de datos se maneja con la librería de ORM para Python llamada SQLAlchemy, la cual a su vez se maneja con ayuda de Flask-SQLAlchemy, una extensión para Flask que ayuda a integrar SQLAlchemy a proyectos hechos con dicho framework. La base de datos en sí es una base PostgreSQL, en parte usada por sobre la base de datos por defecto de Python (SQLite) por proveer más de funcionalidad (por ejemplo, soporta listas y JSON como campos de entidades de la base de datos).

4.5.2. Modelo de la Base de Datos

Se modela una base de datos para soportar las funcionalidades del prototipo y ayudar a identificar los datos necesarios para la implementación del sorteo verificable. En esa misma línea, cabe notar que el objetivo de la base de datos hecha no es simular exactamente lo que hay en el ambiente real, sino que ayudar en el prototipo de, específicamente, el sorteo verificable.

A continuación, se listan los modelos implementados en la base de datos. Se indica cuáles modelos representan datos que se han supuesto ya existen en el ambiente real (no necesariamente en la misma forma en que aquí se implementan) y cuáles serían nuevos.

1. **Persona:** se ha supuesto que en el ambiente real existe. Es necesario tener esto, pues la ciudadanía debe poder identificarse en el sistema para poder obtener información pertinente a cada persona. Además de sus relaciones con otros modelos, se le dan solo campos identificadores (RUN, dígito verificador, nombres y apellidos) los cuales son fácil suponer ya son datos que el SERVEL almacena.
2. **Mesa:** se ha supuesto existe. Representa las mesas receptoras de sufragio. Aparte de su información identificadora (nombre de la mesa), sus campos más notables son aquellos que lo relacionan con otros modelos.

3. **Nomina:** se ha supuesto existe. Representa a la nómina de candidatos a vocal de mesa para una mesa en particular. Es esencialmente una tabla de relación entre **Mesa** y **Persona**, pero además tiene un campo que indica cuándo fue la última modificación realizada. Si se obtiene un pulso de aleatoriedad que haya sido generado antes de la última modificación a cualquier nómina, eso da paso a potenciales manipulaciones, por lo que se incluye ese campo para poder validar contra esa posibilidad.
4. **VocalMesa:** se ha supuesto existe. Es una tabla de relación entre **Mesa** y **Persona** que representa que una persona ha quedado designada como vocal para esa mesa al final de todo el proceso. Se necesita esta tabla porque los resultados del sorteo en sí no necesariamente implican que una persona se designa como vocal, sino que dan una lista de potenciales vocales de mesa, que pasarán a ser designados de ser necesario
5. **Circunscripcion:** se ha supuesto existe. Para los propósitos del prototipo, el punto más importante de esta tabla es que es parte del camino de relaciones por el cual se llega desde una persona a su junta electoral (una persona pertenece a una circunscripción, la cual está bajo el cargo de alguna junta electoral).
6. **JuntaElectoral:** se ha supuesto existe. Contiene la información de las juntas electorales. Además de su nombre, tienen una relación directa de una a muchas con las circunscripciones, significando las circunscripciones bajo el cargo de la junta. Se les agrega además un campo booleano que indica si participan en la prueba del sorteo verificable o no.
7. **Sorteo:** se ha supuesto que ya existe una representación de la información para sorteos normales en el ambiente real, pero se agregó a esto información sobre sorteos verificables. Tiene datos como los números que han salido sorteados, así como una relación con las juntas electorales, lo que expresa que es el sorteo llevado a cabo por cierta junta. Además de un campo de relación al modelo **PulsoSorteo**, se añade una distinción entre sorteos normales y sorteos verificables, usando el tipo de columna **ChoiceType** dado por la librería **SQLAlchemy-Utills**.
8. **PulsoSorteo:** nuevo. Representa la relación entre una entidad **Sorteo** y una entidad **Pulso**. La información sobre el número de reintentos realizados durante el sorteo se deja aquí.
9. **Pulso:** nuevo. Es la información de un pulso de aleatoriedad emitido por el faro del CLCERT. Además de su relación a la tabla **PulsoSorteo**, tiene toda la información de un pulso de aleatoriedad.

Con estos modelos, se tienen las relaciones e información que permiten construir lo buscado con el prototipo. Más que la implementación final exacta, se apunta a que con esto el prototipo demuestre qué información se necesita. La forma en la que los datos se representan probablemente cambiaría en el ambiente real. Un ejemplo de esto último es el uso de columnas tipo "ARRAY", las cuales se pueden usar con **SQLAlchemy** cuando se tiene una base en **PostgreSQL**; en vez de listas directas, el ambiente real podría usar tablas intermedias.

4.5.3. Petición de Pulso de Aleatoriedad al Faro

Usando el servidor y la base de datos, se crean servicios web con el propósito de abrir puntos de acceso y manejo de los datos para el frontend implementado. Se ha supuesto tam-

bién que se tiene un sistema de autenticación, el cual permite identificar a distintas juntas electorales que realizan peticiones.

De los servicios web, se considera importante explicar cómo es el acceso al pulso a usar en el sorteo: se idea un flujo en donde el backend actúa como mediador del SIVOME, obteniendo y registrando el pulso. Este flujo se muestra en la figura 4.3, y se discute también en la sección 4.6.2 (allí más en términos del frontend).

EL servicio web creado se encarga de obtener y pasar un pulso de aleatoriedad. Se han implementado dos métodos para realizar la petición desde el servidor al faro de aleatoriedad. En cualquiera de los dos casos, luego de que el pulso se obtenga correctamente por primera vez, éste queda registrado y asociado a juntas electorales según corresponda. Mientras éste sea válido, será el pulso que se retorne en futuros llamados del servicio realizado por las juntas electorales que lo tengan asociado.

El primer método para procurar el pulso, que es en especial útil para ayudar a probar el prototipo, es que el backend se lo solicite al faro cuando le llegue una petición por un pulso. Esto quiere decir que el pulso se consigue cuando un usuario lo demande por primera vez y no antes. En términos de frontend, esto sería que el pulso se obtiene desde el faro solo cuando una junta electoral accede a la página de sorteo verificable del SIVOME por primera vez.

El segundo método es programar una fecha a partir de la cual se intentará obtener el primer pulso sin errores que el faro genere a partir de ese momento. En este caso, no hay una junta electoral específica que se pueda asociar con el pulso, sino que éste se asocia a al sorteo verificable de todas las juntas electorales que participen del sistema verificable.

En ambos casos, existe una consideración de seguridad: si se permite que las juntas electorales confeccionen una nómina de candidatos para la cual se usará un pulso de aleatoriedad ya existente, entonces existe una posibilidad de manipulación. En concreto esa nómina se pudiera confeccionar sabiendo qué números se derivan del pulso, lo que facilitaría escoger directamente a algún candidato.

El SIVOME solo permite acceder a la página del sorteo una vez que todas las nóminas estén terminadas, pero incluso con esto, en teoría se podría terminar la última nómina en el mismo minuto en que se realiza la petición del pulso de aleatoriedad, lo que implica que el pulso se generó antes de terminar todas las nóminas. Entonces, para prevenir el uso de un pulso ya existente, el procedimiento a la hora de hacer la petición al faro debe ser el siguiente:

1. Calcular tiempo t_0 , correspondiente a la última vez que la junta electoral hizo cualquier modificación a cualquiera de las listas de candidatos a vocales de mesa de las que son responsables.
2. Calcular tiempo t_1 , correspondiente a cuándo empieza el primer minuto después del minuto cuando ocurre t_0 . El faro de aleatoriedad genera un pulso al inicio de cada minuto, por lo que t_1 indica cuándo se puede pedir el primer pulso que no pudo haber influenciado la confección de las nóminas de candidatos.
3. Obtener el tiempo actual t_a .
4. Si $t_a > t_1$, hacer una petición para obtener un pulso de aleatoriedad. Si no, esperar (por lo menos) $t = t_1 - t_a$, y luego pedir el pulso.

En la práctica las nóminas ya están listas con amplio tiempo antes del sorteo. Pero, en el caso extremo de que se necesite esperar, de todas maneras el tiempo t no será más de un minuto, por lo que se considera factible hacer esperar a los usuarios esa cantidad de tiempo.

El primer modo de obtener un pulso, si bien más flexible, en principio abre la posibilidad de que se manipule el resultado del sorteo de una manera similar, debido a que se puede elegir el momento en el cual se usa el faro. Como los pulsos se generan cada un minuto, teóricamente, antes de que se acabe el minuto, se podría ver el valor aleatorio del pulso (por fuera del SIVOME) y simular el sorteo antes de realizarlo realmente. Así, se podría esperar al siguiente pulso y ver si éste da resultados más deseables antes de si quiera acceder a la página del sorteo.

Para contrarrestar esa posibilidad, lo que se hace es atrasar el pedido hasta que comience el siguiente minuto. Entonces, necesariamente se consigue un pulso cuyo valor le era desconocido al usuario. Al mismo tiempo, como efecto secundario, se resuelve el problema del tiempo de confección de las nóminas antes mencionado.

Un lado negativo del primer método es que fuerza a usuarios a esperar por el pulso. Otro problema, es que es relativamente natural cuestionar si efectivamente se implementa dicho retraso de la petición. Esto puede pasar porque terceras partes verificadoras podrían percibir a las juntas electorales y al SERVEL como una sola entidad, de modo que no confían en que SERVEL realmente obligue a las juntas electorales a actuar de cierta forma o bajo ciertas restricciones.

Una manera de responder a esto sería que las juntas electorales, individualmente, declaren un momento en el que pedirán el pulso. Eso implica que las terceras partes pueden saber de antemano que hubo un acuerdo sobre cuándo pedir el pulso, con lo que entienden que no se podía hacer el sorteo con conocimiento del valor aleatorio. Llegar hasta este punto, sin embargo, hace natural pensar en el segundo modo de obtener el pulso: programarlo para una hora y fecha en específico, en lugar de dejar a cada junta decidir cuándo pedirlo.

En cuanto a la habilidad de convencer a terceras partes de que no hay manipulación de los resultados, se prefiere que se establezca un momento específico en el cual obtener el pulso de aleatoriedad.

Dado que la ley especifica un día y hora para el sorteo de vocales de mesa (definidos en relación al día de realización del proceso electoral asociado), se considera que tiene sentido práctico que el backend programe un momento exacto para procurar la aleatoriedad verificable. Se puede entonces, por ejemplo, intentar pedir el pulso a partir de 5 minutos antes de la hora del sorteo (con lo que se podría evitar a los usuarios el tiempo de espera).

En el prototipo del servidor web, se crea una configuración que permite cambiar entre conseguir el pulso según se acceda a la página de sorteo verificable y que el servidor programe pedirlo en un momento en específico. El primer modo aquí se designa más bien como una configuración de desarrollo, mientras que el segundo, acordar de antemano una fecha y hora, es el fuertemente recomendado para el ambiente real, si bien puede resultar más difícil lograrlo, pues depende de que pueda llegarse a ese acuerdo.

El servicio web de este backend que entrega el pulso a la página de sorteo además entrega la semilla asociada al pulso directamente (es decir, el resultado de combinar el valor aleatorio del pulso con el nombre de la junta por medio de SHA-3 512). Si la petición es hecha por una junta electoral que ya tiene un pulso, y si no ha modificado ninguna nómina de candidatos desde entonces, se le entrega ese mismo pulso que la junta electoral ya tenía asociado. En ese caso también se retorna la cantidad de reintentos que la junta electoral tiene registrada para su sorteo, para poder retomar el sorteo donde se hubiese dejado la última vez.

4.5.4. Otros Servicios Web

Además de conseguir y registrar el pulso de aleatoriedad, se implementan otros servicios web que apoyan y dan funcionalidad al prototipo. Entre ellos están:

1. Registrar reintentos: para el sorteo verificable, se abre un servicio por medio del cual el frontend debe registrar reintentos del sorteo verificable apenas ellos sucedan.
2. Sortear: se crean APIs para registrar el sorteo, tanto manual (el sistema original) como verificablemente. El servidor recibe peticiones que vengan con datos de algún sorteo. Si los datos son válidos, el servidor los registra y finaliza el sorteo. En el caso del sorteo verificable, se verifica que dado el pulso asociado a la junta electoral, se tengan los mismos resultados que los pasados en la petición. Para la recreación del sorteo, se usa la versión para Python de la librería descrita en el capítulo 4.3 (adaptada por Camilo Gómez [26]).
3. Información para Consulta SERVEL: recibe el RUN de una persona y da la información sobre el sorteo de vocales de mesa asociado a ella. En primer lugar, se da información sobre su asignación a vocal de mesa (si se le asignó como vocal por medio de un sorteo verificable, el sorteo original o por otro medio). Si hubo sorteo verificable, el servicio incluye el pulso de aleatoriedad en su respuesta, así como un vínculo al certificado del pulso emitido por el CLCERT (como el visto en la figura 2.3). Este servicio está pensado para mostrar la información adicional que la página de Consulta de Datos Electorales del SERVEL podría necesitar de implementarse el sistema verificable. Se discuten más en detalle las modificaciones propuestas a esta página en la sección 4.7.
4. Información para nueva página donde se detalla el sorteo verificable: se propone una página donde se detallen y expliquen los detalles del sorteo verificable, la cual es discutida en más detalle en la sección 4.7.3. Para demostrar los datos que esta página necesitaría, se implementa un servicio que, dado el RUN de alguna persona, responde con los datos asociados al sorteo verificable relevante para esta persona. Estos datos son: el pulso de aleatoriedad; la nómina a candidatos de vocales de mesa, para la mesa receptora de la persona consultante, incluyendo la nómina ordenada alfabéticamente y un timestamp de la última modificación hecha en la nómina; el número de reintentos que hubo durante el sorteo, y el nombre de la junta electoral. Con todo lo cual se puede rehacer el sorteo de vocales de mesa.
5. Verificador de pulsos: se crea un servicio por medio del cual se le puede pedir al servidor correr los scripts de verificación de pulsos del CLCERT y que responde con un JSON que contenga los resultados. Los scripts se modifican de modo que sea una función que se pueda importar y llamar, junto a que en lugar de solo imprimir los errores en pantalla se

retorne información sobre estos. Este servicio está pensado para ser usado por terceras partes. En la sección 4.7.5 se habla sobre la implementación de una aplicación web simple que utiliza este servicio para verificar.

Con la excepción del verificador de pulsos, la mayoría de los servicios mencionados están pensados para ser usados por el SIVOME o páginas del SERVEL, lo que implica que son de uso interno para el software desarrollado por el SERVEL. Todos estos servicios son activados por peticiones HTTP y, generalmente, responden con JSON.

Además de lo mencionado en la lista, se crea una página con un resumen de los detalles técnicos del sorteo verificable. Serviría para que personas con conocimiento técnico vean los datos de un sorteo verificable en un formato resumido y general, en contraste a la página mencionada en el ítem 4 de la lista anterior, que da explicaciones extensas y también da información sobre la mesa receptora del visitante. Por simplicidad, en su implementación, esta página es servida completamente por el servidor, aprovechando el sistema de plantillas (*templates*) de Flask para pasar la información contenida en la base de datos. Se habla más de esta página en la sección 4.7.4.

4.6. Sistema de Selección de Vocales de Mesa

4.6.1. Plan de Intervención

La existencia y uso del SIVOME, descrito en la sección 2.2.3 en el Marco Teórico, abre la posibilidad de crear software que las juntas electorales pueden estar considerablemente dispuestas a usar, dado que se podría implementar dentro de una aplicación ya familiar para ellas.

Para tener un sistema de selección que sea capaz de usar aleatoriedad verificable, lo que se hace entonces es intervenir en el SIVOME, tanto en el frontend de la aplicación web como en el backend donde estén los datos usados y registrados.

Dado que el SIVOME ya tiene un paso para registrar los 10 índices que se eligen al azar, hay una oportunidad para añadir la selección por medio de aleatoriedad verificable ahí, de forma de aprovechar que las juntas electorales ya están familiarizadas con el uso de un cierto software para hacer su trabajo. Se busca con esto que el cambio para las juntas electorales sea lo menos brusco posible y que el uso del sistema de selección verificable sea así de su comodidad.

Para implementar el cambio, se buscan juntas electorales que acepten una invitación a participar en la prueba de un prototipo, y se busca también obtener la aprobación y cooperación del SERVEL para implementar dicho prototipo. El objetivo de esto es no forzar a todas las juntas electorales a participar, en especial no de forma inesperada para ellas. Dado que para acceder al SIVOME se necesitan credenciales, se puede distinguir entre las distintas juntas electorales dentro del sistema y **ofrecer** acceso a las nuevas funcionalidades a esas juntas. Énfasis en “ofrecer” pues se estima importante que el cambio sea opcional, de forma

de tratar de acomodar a los deseos de las juntas electorales lo más posible y evitar reticencia al cambio.

Durante el trabajo de memoria se hicieron reuniones con el SERVEL lo que ayudó a orientar el trabajo (por ejemplo, consiguiendo una explicación de lo que es y hace el SIVOME, incluyendo obteniendo el manual referenciado en la figura 2.1) e incluso a obtener consejos sobre qué junta electoral sería potencialmente la más dispuesta a participar. A raíz de esto último, se contactó a la junta electoral de Macul, quienes se mostraron interesados en la idea y también ayudaron dando su perspectiva y clarificando ciertas dudas que se tenían.

Los miembros de la junta electoral de Macul también están interesados en agregar más datos sobre la ciudadanía al SIVOME, de forma de ayudarlos a estar mejor informados para seleccionar candidatos, lo cual es considerado bueno no solo porque da una oportunidad de mejorar el sistema más allá de lo originalmente planeado, sino que también ayuda a tener algo que ofrecer de vuelta a las juntas electorales por su ayuda.

Por otro lado, aún no se han hecho acuerdos propiamente formales con las juntas electorales o el SERVEL, si bien los procesos están en camino. Es en parte por esto que tampoco se ha podido acceder al código del SIVOME, y por lo tanto durante el trabajo de memoria se presentan mockups o prototipos que no están insertos en el software real del SERVEL.

Más concretamente, el plan que se sigue es: crear un mockup de la página del SIVOME antes mostrada en la figura 2.1, pero a éste agregarle una opción para acceder a una nueva versión del proceso de selección. La nueva opción se parecería mucho a la página original, pero con la interfaz algo cambiada para reflejar la nueva funcionalidad, y asegurando que haya una forma de volver a la versión original del sorteo. Todo esto siguiendo la línea de no querer causar un cambio demasiado brusco y que todo pueda ser opcional.

The image shows a web interface mockup for a voting process. At the top, there is a header with the title "Sorteo Vocales" on the left and a link "Usar Sorteo Nuevo" on the right. Below the header, there are two main sections. The first section is titled "Sorteo de Vocales Titulares" and contains five numbered input boxes labeled "#1" through "#5". The second section is titled "Sorteo de Vocales Suplentes" and also contains five numbered input boxes labeled "#1" through "#5". At the bottom of the page, there is a button labeled "Finalizar" and a link labeled "Finalizar Sorteo Vocales".

Figura 4.2: Recreación de la página del SIVOME donde se realiza el sorteo, pero donde se ha añadido un vínculo para acceder a la versión nueva del sorteo.

Según se pudo observar del manual provisto por el SERVEL, la página parece tener estilos similares a los usadas por el framework CSS “Bootstrap”, por lo que se usa esto para ayudar con la recreación de la página. La página original activa el botón de “Finalizar” solo una vez que se hayan escrito 10 números válidos, así que también se recrea eso, y por eso en la captura de pantalla la última sección tiene un estilo que la hace parecer desactivada, la cual cambia al escribir los 10 números. Se usa Vue.js, framework de JavaScript para construir páginas reactivas, para poder rápidamente crear prototipos con la funcionalidad de validar los números escritos y cambiar el estilo de la última sección cuando fuese adecuado.

Si se hace clic en “Usar Sorteo Nuevo” se llega al prototipo de la nueva versión del sorteo.

4.6.2. Nuevo Sorteo

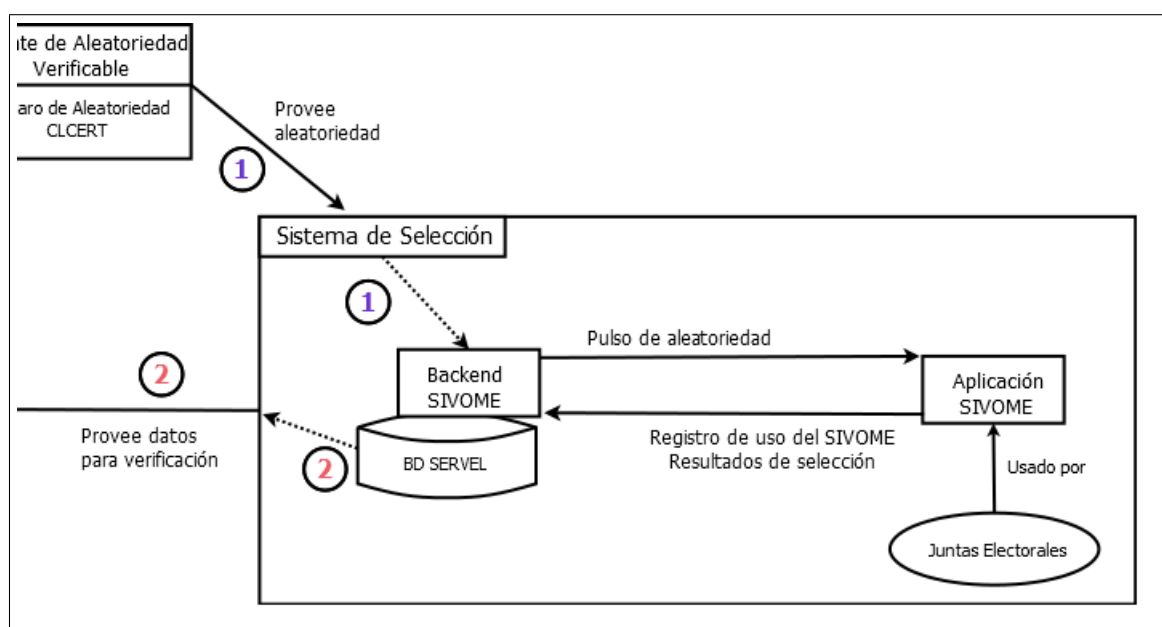


Figura 4.3: Más detalle sobre el sistema de selección, formateado como un “acercamiento” del diagrama de la figura 4.1.

En términos generales, el sorteo nuevo sigue las ideas presentadas en la figura 4.3.

A través del frontend del SIVOME se pedirá un pulso, pero esto es mediado por el backend que es quien lo pide al CLCERT. Con este flujo se tiene que el backend puede registrar directamente el pulso que la junta electoral debería estar usando. Las ventajas de hacerlo de esta forma son que así se le puede asignar un pulso en particular a la junta electoral para esa selección y se puede persistir asociándolo a la cuenta de ella, dando así una forma de evitar que, por ejemplo, una recarga de la página o un cambio de navegador/computador usen un pulso completamente nuevo. El punto de esto a su vez es que de otro modo se podría, en teoría, reintentar la selección con distintos pulsos sin que hubiese un registro de esto, lo que implica que hay una vulnerabilidad muy simple de aprovechar para manipular los resultados de la selección sin que la verificación luego pueda revelar realmente el proceso completo. En la otra dirección, el SIVOME envía datos relativos a cómo la junta electoral usa la aplicación,

no solo para obtener los resultados finales, sino que además a modo de que luego los sistemas de verificación puedan reconstruir la selección.

En las siguientes secciones se muestran capturas de pantalla del prototipo donde se ejemplifica lo descrito en el párrafo anterior y se explica cómo funciona la página de selección.

Nuevo Sorteo: Cargando

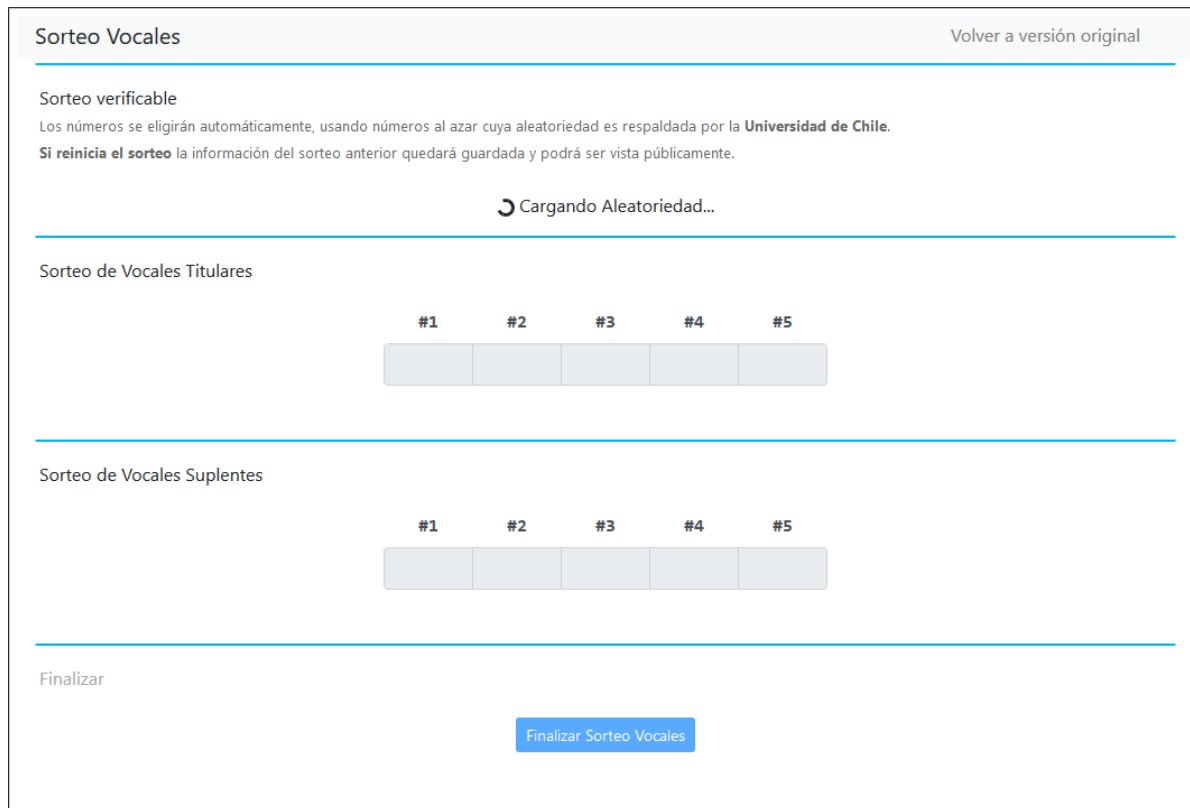


Figura 4.4: Prototipo para página de sorteo verificable para SIVOME, donde se está esperando obtener el pulso de aleatoriedad.

En la figura 4.4 se muestra lo primero que se ve al abrir la página del sorteo verificable. Con respecto a la versión original (vista en la figura 4.2), se ha añadido una nueva sección encima de las dos originales, en la que está todo lo directamente relacionado con la nueva forma de realizar la selección. Para el caso de esta figura, lo que se muestra es un mensaje de “Cargando”, que cuando termine el proceso será reemplazado por los controles del sorteo (mostrados en las siguientes secciones, por ejemplo en la figura 4.5). Además de dicho mensaje, se muestran mensajes sobre el sorteo verificable así como una advertencia sobre los reinicios.

La carga de aleatoriedad corresponde a que la página envíe una petición al backend para obtener un pulso de aleatoriedad, Por lo explicado en 4.5.3, esto puede tardar y ser retrasado a propósito hasta aproximadamente un minuto.

También como se explica en la sección 4.5.3, se implementan dos formas de obtener el pulso. Si se está usando el método donde el servidor pedirá un pulso para una junta electoral

solo una vez que ella acceda a la página del sorteo verificable, entonces en este punto el backend usa la API del faro de aleatoriedad para pedir el último pulso emitido. Si, por otro lado, se usa el método donde el backend programa obtener el pulso en una fecha y hora exactas (o si no es la primera vez que la junta accede a la página del sorteo), entonces el servidor responde con el pulso que la junta electoral ya tenga asociado a su sorteo verificable.

Que la petición pase por el servidor, y así se tengan estos vínculos entre sorteos y pulsos de aleatoriedad, permite recobrar los datos de la selección si, por ejemplo, la página del sorteo es accidentalmente recargada en medio de la selección, lo que es una buena ventaja tanto en términos de usabilidad como seguridad. En usabilidad, si se accede a la página de nuevo no hay que esperar a obtener el pulso del faro, pues el servidor ya lo tiene. En seguridad es también importante tener estos vínculos mediados por el servidor, porque si la petición del pulso la hiciera directamente el frontend al faro de aleatoriedad, sería más fácil manipular el sorteo para que se use un pulso de aleatoriedad que de resultados específicos.

En términos de estética, la página del sorteo verificable es, además de la sección nueva, casi igual a la original, con la excepción de que los campos de *input* se han deshabilitado, pues es la aplicación la que se encargará de escribir cada número. También, al igual que en el SIVOME original, la sección de “Finalizar” tiene un estilo que refleja que la selección no se ha completado.

Nuevo Sorteo: Seleccionando

Cabe enfatizar que, un aspecto importante de usar el servidor como mediador entre el frontend y el faro es el registro del proceso de selección. Es decir, se podría permitir reintentar la selección, pero siempre y cuando esto se haga transparente al sistema de verificación, lo que puede ser útil si sucede algún problema o algún error durante la primera selección que lleve a la junta electoral a, por cualquier motivo, querer reintentarla con nuevos números.

En el prototipo hecho, se permiten reintentos con nuevos números, generados a partir del mismo pulso que los anteriores, pero se lleva cuenta explícita de cada reintento. Esto se puede ver en la figura 4.6.

Parte del motivo por el que se permiten reintentos es también que prohibirlos completamente podría potencialmente percibirse por las juntas electorales como demasiado restrictivo, por lo que en primera instancia no se restringe de esa forma para minimizar la posibilidad de rechazo. Si bien esto puede parecer sospechoso a terceras partes, se da la posibilidad a las juntas electorales de elegir hacer esto o no, sabiendo que quedarán registros de sus acciones (lo que, en cierto sentido, es en sí una manera de disuadir el uso de esta funcionalidad).

Para obtener los números se utiliza un generador de números pseudoaleatorio basado en el cifrador de flujo ChaCha20. La forma en la que se generan y usan los números pseudoaleatorios está explicada más en detalle en la sección 4.3. Como se detalla en 4.2.3, la semilla dada al generador corresponde a los primeros 320 bits resultantes de hacer hash, por medio de SHA-3 512, entre el valor aleatorio del pulso, encontrado en su propiedad `outputValue`, y el nombre de la junta electoral.

Sorteo Vocales
Volver a versión original

Sorteo verificable
 Los números se elegirán automáticamente, usando números al azar cuya aleatoriedad es respaldada por la **Universidad de Chile**.
Si reinicia el sorteo la información del sorteo anterior quedará guardada y podrá ser vista públicamente.
Identificador del pulso aleatorio: 445646

Siguiente número
Completar todo automáticamente
Reiniciar

Sorteo de Vocales Titulares

#1	#2	#3	#4	#5
23	14			

Sorteo de Vocales Suplentes

#1	#2	#3	#4	#5

Finalizar

Finalizar Sorteo Vocales

Figura 4.5: Prototipo para página de sorteo verificable para SIVOME. En esta captura se han seleccionado ya los primeros dos números.

En la figura 4.5 se puede ver la página ya cargada, donde se han seleccionado los primeros dos números. Además se puede ver como los controles han reemplazado al mensaje de carga. Se colocan allí los siguientes botones:

1. Siguiente número: simplemente rellena el siguiente campo disponible con un número seleccionado al azar.
2. Completar todo automáticamente: completa todos los campos restantes en una rápida secuencia.
3. Reiniciar: reinicia la selección. Inhabilitado hasta que se obtenga por lo menos un número.

Se tiene el botón 1 porque el sorteo actualmente es un evento donde comúnmente los números se obtienen uno a uno a través de una tómbola, por lo que hacerlo usando este botón puede resultar más natural. Se tiene también el botón 2 para ofrecer una opción más instantánea (de todos modos, los números aparecen automáticamente uno a uno, con relativa rapidez). Similarmente, pensando en que puede haber espectadores, se muestra también el número identificador del pulso en la página, de forma que pueda ser visto o pedido, permitiendo potencialmente verificar la selección en tiempo real.

Si se usa el botón de reinicio, primero se pide una confirmación de que de verdad se desee cancelar el sorteo actual. Si se confirma el reinicio entonces se realiza una petición POST

al servidor, con la que se le avisa al backend que debe registrar un reinicio en los datos del sorteo llevado a cabo por la junta electoral.

De completarse la petición correctamente, se borran los números seleccionados, de forma que la siguiente vez que se intenten rellenar se tengan nuevos números. Los siguientes números que se generan son como si se hubiese hecho una selección completa con el pulso, y luego se comenzara otra nueva, es decir, al momento de realizar la verificación y para obtener los números correctos del generador, cada reinicio implica realizar una selección completa, descartar los números, y luego realizar nuevamente el sorteo, usando siempre el mismo pulso. Además de todo esto, se registra también visualmente el número de reintentos del sorteo en la página misma, como se puede ver en la figura 4.6.

Sorteo Vocales Volver a versión original

Sorteo verificable
Los números se elegirán automáticamente, usando números al azar cuya aleatoriedad es respaldada por la **Universidad de Chile**.
Si reinicia el sorteo la información del sorteo anterior quedará guardada y podrá ser vista públicamente.
Identificador del pulso aleatorio: 445646
Número de reintentos: 1

Siguiente número Completar todo automáticamente Reiniciar

Sorteo de Vocales Titulares

#1	#2	#3	#4	#5
21	25	26	7	11

Sorteo de Vocales Suplentes

#1	#2	#3	#4	#5
30	12	5	4	10

Finalizar

Finalizar Sorteo Vocales

Figura 4.6: Prototipo para página de sorteo verificable para SIVOME, donde se ha usado el botón de reinicio una vez.

Nuevo Sorteo: Finalización

Una vez se hayan rellenado los 10 campos, se activa la página de finalización, como se puede en la figura 4.6 donde el botón se ha habilitado. Al presionar dicho botón, se pregunta por confirmación a través de un modal, que es como se hace en el SIVOME real, y que se ha implementado tanto para la versión nueva de la selección como para la copia de la página original. En la figura 1 en la sección de Anexos puede verse una porción del manual donde

se ve el modal de confirmación original, en la figura 2 se ve la copia implementada de esto, con ligeros arreglos y cambios al lenguaje usado.

Luego de confirmar, el frontend realiza una petición PUT al servidor, donde envía los siguientes datos:

- Índices seleccionados.
- Número de reintentos.
- Pulso de aleatoriedad usado.

El servidor a su vez, confirma los datos independientemente, esto es: confirmar que el número de reintentos mandado coincida con lo que se tiene registrado en la base de datos, hacer lo mismo con el pulso y, finalmente, verificar si con el pulso asociado a la junta electoral que hizo la selección, y con ese número de reintentos, entonces se obtienen esos índices. Si no, se retorna un error al frontend y este muestra que hubo un error con la validación de los datos. En caso de validarse los índices, entonces estos se guardan en la base de datos y se responde al frontend que la validación fue correcta (un simple mensaje de “OK”). El sorteo queda marcado como finalizado, por lo que todas las opciones en la página se deshabilitan.

Ya que se hayan validado los datos, la página de selección simplemente muestra un último mensaje de éxito, como se puede ver en la figura 4.7. Se sigue un estilo similar al resto de la página, usando un modal de Bootstrap para el mensaje, y ya que el mensaje mismo está sobre los campos con los números, también se escriben en este los números seleccionados.

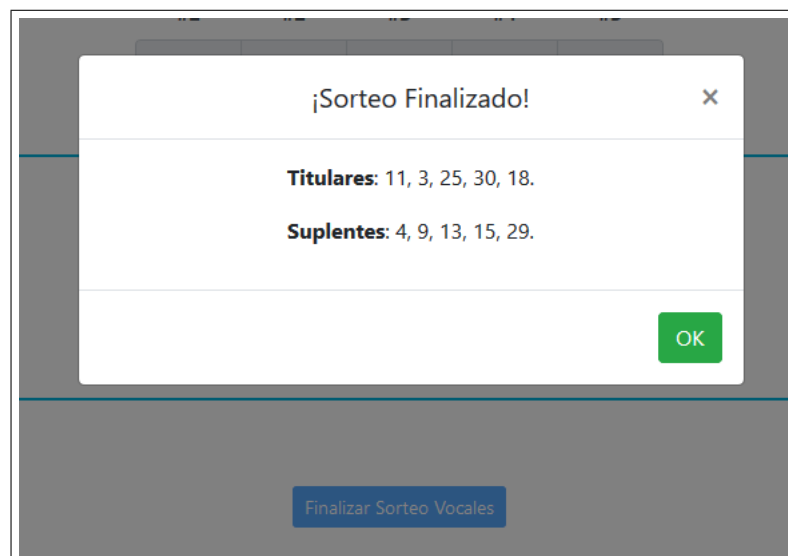


Figura 4.7: Modal para confirmar la finalización de la selección en el prototipo para página de sorteo verificable.

4.7. Sistema de Verificación Sorteo de Vocales de Mesa

4.7.1. Plan de Intervención

De lo descrito en la sección 2.2.4 en el Marco Teórico, es fácil notar que la información de los detalles está de cierto modo restringida, pues para investigar sobre más detalle que solo los resultados de la selección se debe acceder físicamente a los registros. En el caso del sorteo, este también es un evento realizado en un espacio y tiempo específicos.

Por otro lado, la existencia y uso del sitio web del SERVEL da una apertura para implementar más puntos de acceso a información sobre los procesos electorarios, en contextos e interfaces que los usuarios ya encuentran relativamente familiares.

La intervención que se planea hacer aquí es añadir información a la página de Consulta de Datos Electorales del SERVEL, para incluir más detalles del sorteo. Junto a esto, se planea mostrar una recreación paso a paso del proceso de selección, para demostrar que fue correcto, así como dar pasos a la ciudadanía a través de los cuales puedan realizar la verificación por su propia cuenta. Se desea hacer esto en las páginas del SERVEL porque ya es una donde la ciudadanía consulta para saber sobre la selección de vocales de mesa, y si bien actualmente la información es sobre los resultados más que sobre el proceso, de todas formas se considera que ahí es el lugar más natural para dar tal información y donde hay más posibilidades de llegar al público general.

Se observa, por otro lado, que la información sobre la aleatoriedad verificable puede ser altamente técnica, de modo que lo que se hace es describir primero en términos generales en la página de Consulta SERVEL, y desde ahí se proveen recursos o vínculos para llegar a una explicación más en detalle. El punto de esto es reconocer que mucha de la información que se quiere dar en realidad puede no ser del interés de gran porcentaje de los votantes. Se trabaja suponiendo que las personas tienen distintos niveles de curiosidad, pero además incluso que las personas tienen distintos niveles de aversión a recibir información que no es de su interés, esto es: si se da toda la información de una vez, hay usuarios que pueden resultar abrumados y el efecto puede ser contraproducente al objetivo de explicar y mantener a la gente informada e interesada.

Se quiere rehacer y mostrar paso a paso la selección en páginas web disponibles a, por lo menos, cualquier persona que haya sido elegida vocal de mesa. Puede muy bien esto ser suficiente verificación para muchos usuarios, pero cabe recalcar que el hecho mismo de explicar la aleatoriedad verificable, de dónde se saca y cómo se usa, abre las puertas a que terceras partes verifiquen por sí mismas la selección, fuera del sitio web del SERVEL. En este sentido, si bien se puede explicar la selección y demostrar en las páginas del SERVEL, el “sistema de verificación” como concepto es en realidad algo que está en las manos de terceras partes completar, y parte importante de lo que se busca entonces es habilitarlas a poder demostrarse a sí mismas que todo es (o no) correcto.

Cabe notar también que el sistema de verificación propuesto lógicamente no busca reemplazar el concepto de que el sorteo sea un evento al que se pueda asistir, ni tampoco la

toma de actas físicas, no solo porque ambos procedimientos se deben realizar así por ley, sino que también porque lo que se propone puede funcionar como una adición a lo ya existente. Por ejemplo, durante el evento del sorteo se puede mostrar como se eligen automáticamente los números en una pantalla. Esto último se sabe es una idea razonable, no solo porque en la reunión con la junta electoral de Macul se tuvo una acogida positiva, sino que también porque de reuniones con el SERVEL se sabe que se había planeado tener un botón para seleccionar con números elegidos por el SIVOME automáticamente (una selección aleatoria por computadora en el sentido más común, no verificable). Es por esto que la idea de que durante el evento del sorteo se elija por máquina se entiende como muy plausible. Por otro lado las actas y registros físicos pueden ser tomados de la misma forma de siempre.

Las adiciones que se buscan hacer, en relación a cómo se puede verificar el sorteo de selección de vocales de mesa, consisten principalmente en añadir explicaciones, datos, y crear recursos relacionados al sorteo verificable. En particular, a la página de Consulta de Datos Electorales, así como con la creación de páginas nuevas en el sitio del SERVEL, que puedan servir para ahondar en el tema y ser más específicos. Finalmente, a modo de demostrar la verificación en su totalidad, se crea también una aplicación web que permita verificar un pulso de aleatoriedad dado que se tenga su número identificador.


Habilitado para Sufragar	SÍ
Local de votación	LICEO CARMELA SILVA DONOSO
Ubicación	AV. PEDRO DE VALDIVIA N° 4862, ÑUÑO A Ocultar mapa
	
DESIGNACIONES	
Vocal de Mesa	SÍ.
Miembro Colegio Escrutador	Si cumpliste la función de miembro de Colegio Escrutador el 20 de noviembre, deberás asistir el lunes 18 de diciembre a ejercer el mismo rol.
SU JUNTA ELECTORAL	
Sede	Macul
Dirección	Gregorio de la Fuente N° 3143, Macul, Santiago de Chile.
Email	jfacuse@notariafacuse.cl
Fono	223478902, 223478900

Figura 4.8: Captura de pantalla de la página de la versión modificada de Consulta de Datos Electorales del SERVEL. En esta sección se muestra la adición de un mapa y de información de la junta electoral correspondiente a quien esté consultando la página.

4.7.2. Modificaciones a Consulta SERVEL

En la figura 4.8 se muestra la primera parte de las modificaciones propuestas para la página de Consulta de Datos Electorales. En esta sección se ve un mapa colapsable, que se ha añadido como parte del mockup (y no es funcional), dado que en el ambiente real ya se tiene la ubicación en buen detalle, pero más importante es la adición de una tabla con información más detallada sobre la junta electoral de la persona que esté visitando la página. Esto último se consideró útil para poder visibilizar más la existencia y trabajo de las juntas electorales, dado que, según lo hablado con tanto personas del SERVEL como de la junta electoral de Macul, una buena porción de la ciudadanía chilena parece suponer que todo el trabajo y responsabilidad de la selección de vocales de mesa recae en el SERVEL.

Además de esto, se añade una sección llamada simplemente “Más Información”. La idea es que, si a quien consulta su información en la página se le designó como vocal de mesa, entonces se le entregan detalles relevantes. Primero se le explica cómo se le designó. Dado que el sistema de aleatoriedad verificable sería opcional, se puede ser vocal de mesa por:

- Se está cumpliendo con el periodo de una designación anterior.
- Se le selecciona mediante el sorteo original.
- Se le selecciona mediante el nuevo sorteo (el caso supuesto en lo que se muestra en las capturas de pantalla).
- Cualquiera de los dos casos anteriores, pero era vocal suplente y un vocal titular se excusó.

Entonces la página mostraría distinto texto dependiendo de cómo se eligió al vocal de mesa, por lo menos en cuanto a si se le eligió con aleatoriedad verificable o no. En la figura 4.9 se puede ver la primera parte de la nueva sección que se quiere agregar, donde hay una explicación a la pregunta “¿Por qué soy vocal?” en la cual se habla a nivel general sobre quienes realizaron la selección y cómo, al mismo tiempo que se indican recursos que se pueden seguir para aprender más.

Por otro lado, se es consciente de que agregar solo información de algo tan específico como lo es el sistema de selección verificable parecería extraño, si es que no fuera de lugar, para el usuario común, en parte es por eso que la explicación es más general que solo eso, pero además esto puede verse como una oportunidad de añadir más datos que pueden ser relevantes para la ciudadanía. Al hablar con la junta electoral de Macul se escuchó de ellos, antes si quiera de mencionar que el plan era añadir información como esta, que sería bueno dar de forma más directa información a vocales de mesa, para empoderarlos al mantenerlos informados de sus opciones y deberes. Un ejemplo conversado fue dejarles explicaciones sobre cómo excusarse en la misma página de Consulta SERVEL; otro fue vincularlos directamente a la “Cartilla de instrucciones para mesa receptora”, que esencialmente contiene las instrucciones sobre cómo ser vocal de mesa. En la figura 4.9 se muestran una lista de excusas², una sección donde se les vincula a la cartilla mencionada donde podrán ver que deben hacer para cumplir con la función de vocal, y una sección general de recursos al final.

²Para el mockup de la nueva consulta.servel.cl: Icono de doctor encontrado en <https://thenounproject.com/term/doctor/22779/>, por Wilson Joseph. Iconos de personas de tercera edad y de persona embarazada encontrados en <https://www.canva.com/>, por Leremy Gan y Icons8 respectivamente.

Más información

¿Por qué soy Vocal de Mesa?

Los detalles legales del proceso pueden encontrarse a partir del Artículo 44, de la [Ley 18.700](#).

En primer lugar, la Junta Electoral de Macul (ex Ñuñoa) le ha seleccionado dentro de la lista de 30 candidatos que se han considerado como los más idóneos para realizar la labor.

En su caso, y según avala el Artículo 46 de la Ley 18.700, a usted luego se le seleccionó para formar parte del grupo de Vocales de Mesa a través de un sorteo.

Los **sorteos realizados por la Junta Electoral de Macul pueden ser verificados digitalmente por cualquier persona** y los números usados han sido **certificados** como efectivamente aleatorios por la Universidad de Chile. Si desea aprender aún más detalles sobre el proceso por el cual se le seleccionó, puede dirigirse a la página [¿Cómo le eligieron de vocal de mesa?](#). Si usted ya conoce el proceso o tiene conocimiento técnico de programación, también hay una página con solo los **detalles técnicos del sorteo** realizado por su Junta Electoral, que puede serle más conveniente como referencia.

Siguientes pasos


Para aprender más sobre lo que debe hacer como Vocal de Mesa, favor de leer la [cartilla de instrucciones para mesa receptora](#).

Las designaciones de vocales de mesa y miembros del colegio escrutador son realizadas por las Juntas Electorales. En caso de preguntas, contactarse con su Junta Electoral.


Por cada acto electoral en el que participe, **se le concederá un bono equivalente a dos tercios de Unidad de Fomento**, aproximadamente \$18.500 (Ley 18.700, Artículo 53). Si además es su primera vez ejerciendo el rol y concurre a la capacitación (ver cartilla para más detalle), se le incrementará el bono señalado en la suma de 0,22 Unidades de Fomento, aproximadamente \$6.000 (Ley 18.700, Artículo 55).

Excusas

De no poder cumplir con su cargo de vocal el día de la elección, **puede presentar por escrito sus causales de excusa ante el Secretario de la Junta Electoral correspondiente, hasta este jueves 2 de noviembre**. De no asistir a cumplir su función sin excusarse, será sancionado con una multa a beneficio municipal de 8 UTM, aproximadamente \$390.000, aplicada por el Juez de Policía Local competente. Puede ver las posibles excusas a continuación.



1: Estar el vocal comprendido entre las causales de inhabilidad contempladas en el artículo 45 de la [Ley 18.700](#). Incluidas en estas causales están: tener cierto tipo de vínculos familiares con candidatos de la elección, ejercer ciertos trabajos gubernamentales, ser no vidente, entre otros. Para más información vease el artículo mencionado.




2: Tener más de setenta años de edad.

3: Haber sido designado miembro del colegio escrutador.

4: Estar física o mentalmente imposibilitado de ejercer la función, circunstancia que deberá ser acreditada con certificado de un médico.

5: Cumplir labores en establecimientos hospitalarios en los mismos días en que funcionen las mesas receptoras, lo que deberá acreditarse mediante certificado del director del respectivo establecimiento de salud.



6: Estar ausente del país o radicado en alguna localidad distante más de 300 kilómetros o con la que no haya comunicaciones expeditas, hecho que calificará la junta.

7: Tener que desempeñar en los mismos días y horas de funcionamiento de las mesas, otras funciones que encomiende esta ley.

8: Estar la mujer en estado de embarazo o de puerperio dentro de las seis semanas previas al parto y hasta veinticuatro semanas siguientes a éste, circunstancia que deberá acreditarse mediante certificado médico, o con la documentación que acredite estar recibiendo el subsidio a que se refiere el artículo 198 del Código del Trabajo.

Recursos

Puede encontrar recursos e información útil en los siguientes vínculos:

- [Preguntas frecuentes sobre las elecciones.](#)
- [Cartilla de instrucciones para mesa receptora.](#)
- [Ley Orgánica Constitucional sobre Votaciones Populares y Escrutinios \(Ley 18.700\).](#)
- [Más detalles sobre la forma en que se le eligió de Vocal de Mesa.](#)
- [Detalles técnicos del sorteo digitalmente verificable hecho por su Junta Electoral.](#)

Figura 4.9: Captura de pantalla de la página de la versión modificada de Consulta de Datos Electorales del SERVEL. Aquí se muestra la adición de una nueva sección con más información relevante a vocales de mesa.

La página enfatiza que el sorteo es realizado por una junta electoral específica, lo cual tiene

dos facetas: primero, ayuda a aclarar que no es el SERVEL el que está encargado de esta parte; segundo, destaca la junta específica que ha optado a usar la aleatoriedad verificable, lo que puede alentar a otras juntas electorales a tomar la misma iniciativa.

Cabe destacar un par de los vínculos mostrados: primero, en la figura 4.9, se ve que hay un vínculo en la palabras que mencionan certificados de aleatoriedad, esto correspondería a un certificado como el mostrado en la figura 2.3 en el Capítulo de Marco Teórico, para el pulso de aleatoriedad que se haya usado en la selección de vocales. Éste puede ser suficiente prueba sobre la fiabilidad del proceso para muchas personas, al ser un documento vinculado a una institución como lo es la Universidad de Chile. El certificado podría considerarse como el primer nivel de verificación a ojos de la ciudadanía.

Si hubiese quienes deseen saber más sobre el proceso, o quienes no quedasen conformes con solo el certificado, luego hay un vínculo en las palabras “¿Cómo le eligieron de vocal de mesa?”. Este vínculo apuntaría a otra página del SERVEL donde se explica más detalle sobre la selección. La sección 4.7.3 trata de esta página. También se puede ver un vínculo hacia una página de “detalles técnicos” sobre el sorteo digitalmente verificable, el cual iría a una página del SERVEL con un resumen general de los datos del sorteo verificable, apuntando a personas con conocimiento técnico. Se habla de ésta en la sección 4.7.4.

4.7.3. Página de Explicación Detallada de la Selección

Si una persona deseara aprender más en detalle sobre el proceso de selección verificable que llevó a que ella fuese elegida vocal de mesa, y sigue los recursos dados en la página de Consulta SERVEL, entonces la idea es llevarla a una página nueva que explique todo más específicamente. En particular, la intención que se tiene es de poder dar herramientas para que la persona seleccionada pueda reconstruir el proceso que la llevo a ser sorteada.

En la figura 4.10, se muestra la primera parte de una página donde se darían las explicaciones mencionadas. El prototipo ahí mostrado se construye usando la página de Consulta SERVEL como base y modificando su contenido, esto pues idealmente esta página formaría parte del sitio web del SERVEL.

Parte de las explicaciones se hicieron con secciones interactivas, así que se usó aquí también, como en la página de selección verificable para el SIVOME, el framework Vue.js para poder agregar reactividad e interactividad.

Para poder lograr una reconstrucción, se debe por lo menos mostrar cómo se llega desde la nómina de candidatos a la lista final de vocales de mesa. Dado que las nóminas pueden confeccionarse y modificarse a través del SIVOME, se puede suponer que es posible acceder a ellas por medio de las bases de datos del SERVEL. Por otro lado, incluso si estas nóminas son de acceso público, la forma en la que se muestren los datos de las personas en las nóminas es un asunto a considerar en sí mismo. Un aspecto de esto es que actualmente las nóminas se tendrían que acceder físicamente, lo que hace que sean públicas en un sentido bastante restringido, mientras que lo que se propone es poner esta información en una página de internet. La diferencia entre esos dos escenarios hace que valga la pena pensar en privacidad.

¿Cómo le eligieron de vocal de mesa?

Para realizar el sorteo de vocales de mesa, las Juntas Electorales siguen los tres pasos que se explican a continuación.

1 - Confeccionar la Nómina

Su Junta Electoral confecciona una lista de 30 personas consideradas por sus miembros como aptas para ejercer el rol, a partir del padrón de su mesa receptora de sufragios. Puede ver la nómina de la que usted formó parte más abajo.

2 - Numeración

La lista se ordena alfabéticamente según nombre completo (apellidos primero), y luego a cada persona se le asigna un número, según su posición en la lista. Con esto, cada persona en la nómina puede identificarse por un número del 1 al 30.

3 - Sorteo

Para la selección, se usan los números asignados en el pasos anterior. La Junta Electoral realiza un sorteo por medio del cual se eligen 10 números al azar, entre 1 y 30, sin repetir ninguno. Ellos se asocian a personas según las numeraciones hechas en el paso anterior. Los primeros 5 números obtenidos en el sorteo corresponden a vocales titulares, mientras que los últimos 5 son reemplazantes. Estos 10 números se usan para las nóminas de todas las mesas receptoras de sufragios bajo el cargo de su Junta Electoral, no solo para la mostrada en esta página, que es únicamente de la mesa receptora que le corresponde a usted.

Para realizar un sorteo que pueda ser verificado digitalmente, y cuyos resultados no puedan ser predichos, los resultados del sorteo se obtienen usando datos aleatorios dados por la Universidad de Chile. En el caso de este sorteo en particular, se tuvieron los siguientes números, en este orden:

25, 22, 24, 19, 21, 4, 2, 1, 5, 29.

Con esto finalmente se obtiene la lista final de vocales titulares y reemplazantes ([hacer clic en la lista de abajo para ver los resultados de estos pasos](#)).

Nómina

- **P. C. Franco A.**
- G. A. Tomás A.
- K. C. Mauricio J.
- Y. F. Damian N.
- C. P. Andrea P.
- D. G. José I.
- Z. B. Gabriel I.
- V. A. Gabriel A.
- C. S. Rodrigo A.
- B. C. Alejandro E.

Figura 4.10: Captura de pantalla de la página que muestra el detalle de la selección de una persona como vocal de mesa.

Un ejemplo de por qué es: si se muestran nombres completos, o el RUN de las persona, entonces cabe la posibilidad de que visitantes de la página sientan hay una violación a su privacidad, al mostrar sus datos tan públicamente.

La manera de afrontar el cómo mostrar los datos de la gente en la nómina aquí es simplemente mostrar los nombres de forma parcial, con el primer nombre completamente visible, pero usando solo las iniciales para el resto de los nombres y apellidos. El nombre parcial de quien visite la página se muestra siempre destacado, como se puede apreciar en las figuras 4.10 y 4.11, para ayudar a que las personas puedan identificarse a sí mismas durante la explicación, incluso si los nombres no se muestran completos.

La página debe tener una forma de identificar a sus visitantes, puesto que debe poder entregar una nómina específica para una persona específica. Puede suponerse que se puede usar el mismo RUN ingresado en Consulta SERVEL (pues es desde ahí que se llega a la página de explicación detallada), donde el vínculo mostrado a la página de explicación detallada se

construye según el RUN ingresado en Consulta SERVEL; o podría también hacerse mediante un campo para ingresar el RUN con reCAPTCHA, al igual que lo hace Consulta SERVEL. En cualquier caso, se ve que se pueden obtener los datos necesarios para la explicación.

25, 22, 24, 19, 21, 4, 2, 1, 5, 29.

Con esto finalmente se obtiene la lista final de vocales titulares y reemplazantes (hacer clic en la lista de abajo para ver los resultados de estos pasos).

Nómina	Numeración	
• P. C. Franco A.	1. A. C. Sergio R. I.	Sorteo
• G. A. Tomás A.	2. B. U. Valentina S.	
• K. C. Mauricio J.	3. B. C. Alejandro E.	
• Y. F. Damian N.	4. B. Ó. Ilana	
• C. P. Andrea P.	5. C. G. Jorge I.	
• D. G. José I.	6. C. S. Rodrigo A.	Vocales titulares:
• Z. B. Gabriel I.	7. C. P. Andrea P.	25: R. A. Alejandro A.
• V. A. Gabriel A.	8. C. H. Valeria L.	22: P. C. Franco A.
• C. S. Rodrigo A.	9. D. G. José I.	24: R. S. Matías J.
• B. C. Alejandro E.	10. D. U. Bastián H.	19: M. S. Cristóbal M.
• C. H. Valeria L.	11. G. O. Benjamín I.	21: O. A. Diego I.
• G. O. Benjamín I.	12. G. A. Gabriel A.	Reemplazantes:
• D. U. Bastián H.	13. G. A. Cristián F.	4: B. Ó. Ilana
• J. M. Joaquín A.	14. G. A. Tomás A.	2: B. U. Valentina S.
• A. C. Sergio R. I.	15. I. C. Patricio A.	1: A. C. Sergio R. I.
• B. U. Valentina S.	16. J. M. Joaquín A.	5: C. G. Jorge I.
• B. Ó. Ilana	17. K. C. Mauricio J.	29: Y. F. Damian N.
• I. C. Patricio A.	18. L. R. Matías I.	
• G. A. Gabriel A.	19. M. S. Cristóbal M.	
• P. N. Darío A.	20. M. C. Thibault B.	
• R. A. Alejandro A.	21. O. A. Diego I.	
• S. Z. Sven	22. P. C. Franco A.	
• L. R. Matías I.	23. P. N. Darío A.	
• M. S. Cristóbal M.	24. R. S. Matías J.	
• M. C. Thibault B.	25. R. A. Alejandro A.	
• G. A. Cristián F.	26. S. Z. Sven	
• C. G. Jorge I.	27. V. C. Rodrigo E.	
• O. A. Diego I.	28. V. A. Gabriel A.	
• V. C. Rodrigo E.	29. Y. F. Damian N.	
• R. S. Matías J.	30. Z. B. Gabriel I.	

Debe notar además que los resultados del sorteo nos dan una lista de potenciales vocales de mesa, a asignar cuando se necesiten, eligiéndoles según el orden en que se obtuvieron sus números en el sorteo. Por ejemplo, puede haber mesas donde a alguien se le haya seleccionado en un proceso anterior, y forme parte de la mesa actual como vocal para cumplir con su periodo designado. En este caso, quedan de vocales para dicha mesa las primeras cuatro personas sorteadas de vocales titulares y aquella que ya estaba designada como vocal de mesa desde antes.

Figura 4.11: Captura de pantalla de la página que muestra el detalle de la selección de una persona como vocal de mesa. De izquierda a derecha se ve como de la lista de 30 candidatos se llega a la nómina con 5 vocales titulares y 5 suplentes.

La lista de nombres³ que se ve en la figura 4.10 se encuentra en una sección interactiva, en la cual se hacen demostraciones de lo que se habla en las tres secciones que se encuentran al principio. Se puede hacer clic sobre la lista, lo que hará que aparezca una segunda lista, esta vez indexada, a la derecha de la primera. A su vez, puede hacerse clic en esta segunda y entonces aparecen los resultados de la selección al hacer la asignación de vocales titulares y reemplazantes según los índices que se obtienen durante el sorteo. Puede verse en la figura 4.11 como queda esta sección al haber desplegado todas las listas. En la figura 3 en la sección de Anexos muestra un ejemplo de cómo se resalta una lista a la que se le puede hacer clic cuando se la apunta con el ratón, a modo de indicar que se puede interactuar con ella.

Con las partes mostradas en las figuras 4.10 y 4.11, se tiene una explicación más detallada de cómo se realiza la selección, que permite ver a grandes rasgos cómo se llega desde la

³Para este ejemplo: nombre resaltado correspondiente al autor de esta memoria, el resto son nombres ficticios y cualquier coincidencia con nombres reales es pura coincidencia.

nómina de candidatos confeccionada por la junta electoral hasta la selección de vocales de mesa, según un sorteo por el que se eligen 10 índices al azar. Hasta este punto, sin embargo, aún no se explica cómo se eligen esos números, por lo que se sigue expandiendo en este punto en las siguientes partes de la explicación, y lo que sigue inmediatamente luego de la demostración con las listas es una sección separada, llamada “Explicación Avanzada” para denotar que se entra ahí en puntos más complejos.

Como introducción a la explicación avanzada, hay un texto que resume lo hablado en ella y ofrece un vínculo a la página de detalles técnicos (de la que se habla en la sección 4.7.4). El texto introductorio se muestra en la figura 4 en Anexos. La explicación avanzada luego se divide en cuatro partes, las cuales se discuten más a fondo a continuación:

1. Recreando el Sorteo en Detalle.
2. Generador Pseudoaleatorio y Semillas de Azar.
3. ¿Cómo se obtiene la semilla de azar?
4. Ejemplo Completo.

Recreando el Sorteo en Detalle

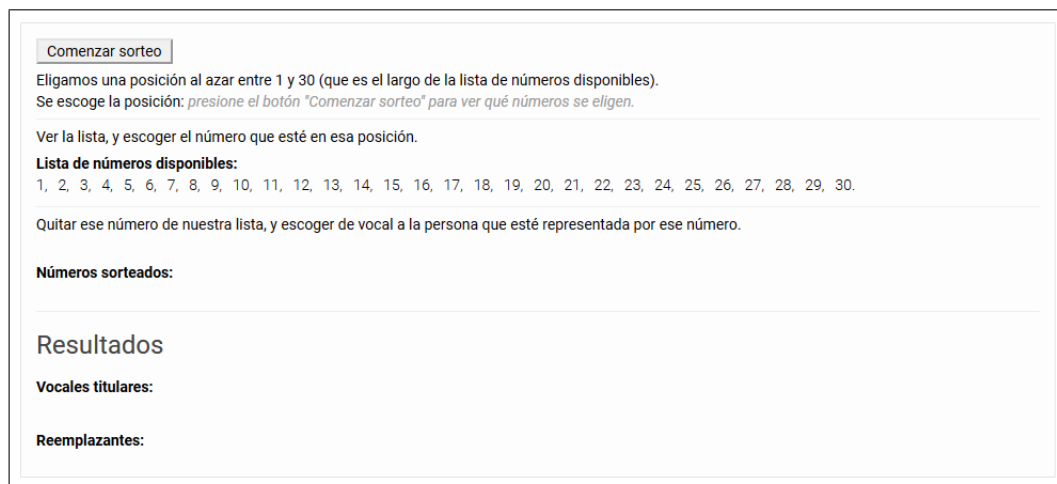


Figura 4.12: Captura de pantalla de explicación interactiva sobre cómo es el proceso del sorteo verificable, en su estado inicial.

En la primera de estas partes, se comienza explicando el algoritmo de selección de forma general, sin ahondar específicamente en generadores de números aleatorios. En la figura 4 en Anexos muestra la explicación en palabras referenciada en este párrafo.

Para poder realmente mostrar cómo se eligen los índices correspondientes a un sorteo en concreto, la explicación luego se vuelve más específica e interactiva, como puede verse en la figura 4.13. En esta parte se muestra un detalle reactivo que permite ver qué valores van tomando las distintas variables del proceso.

La sección interactiva se acompaña también con una vista de los resultados del proceso, lo que puede verse en las lista vacías al final de la figura 4.12, o en la figura 4.14 donde se

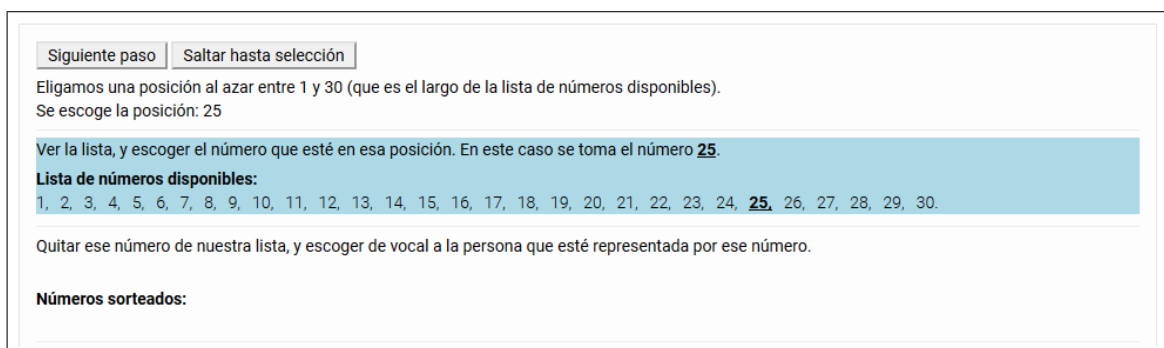


Figura 4.13: Captura de pantalla de explicación interactiva sobre cómo es el proceso del sorteo verificable. En la captura se está en el segundo paso de la primera iteración del algoritmo de sorteo verificable.

muestra como ellas van cambiando y construyéndose mientras el algoritmo avanza.

Los controles son simples botones que permiten: comenzar el proceso (botón “Comenzar”, puede verse en la figura 4.12), avanzar un paso del algoritmo (botón “Siguiete paso”) y avanzar hasta el final de una iteración del algoritmo (botón “Saltar hasta selección”).

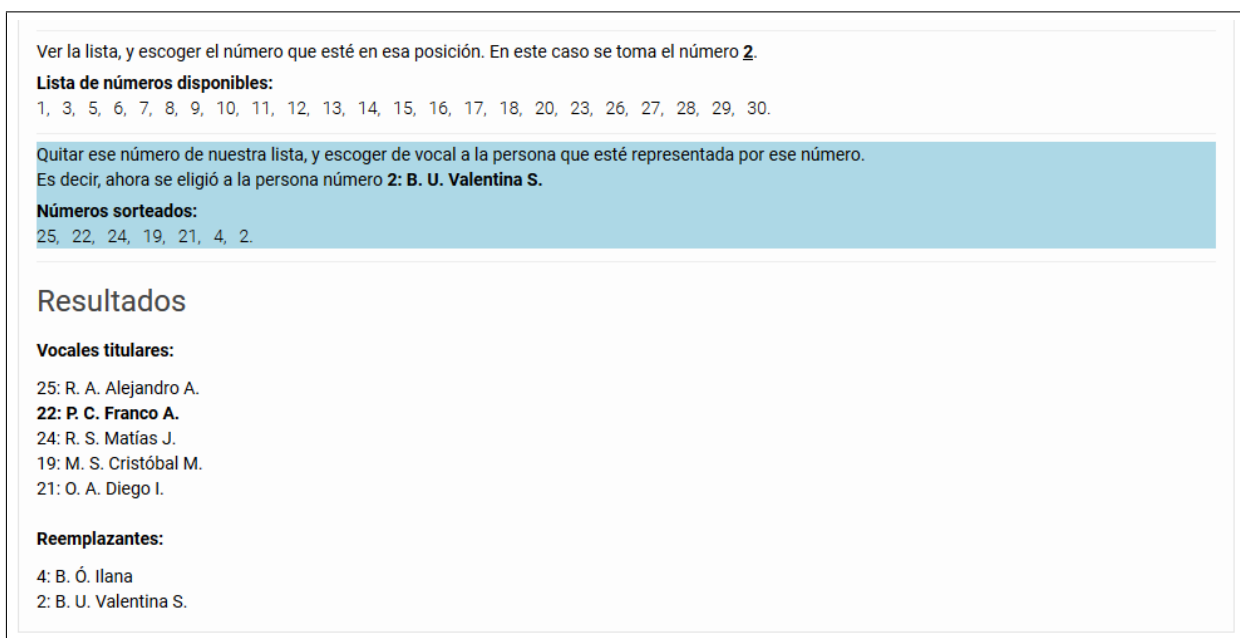


Figura 4.14: Captura de pantalla de explicación interactiva sobre cómo es el proceso del sorteo verificable. Muestra la segunda parte donde se ve cómo van cambiando las listas.

El objetivo de todo esto es habilitar a lectores de la página a poder reconstruir la selección y ayudarles a ver cómo ésta se desarrolla. Para ello se les deja controlar la velocidad del proceso y se les muestran los resultados del algoritmo en distintas formas, como se ve en la figura 4.14. Aquí se ven: la lista de números del 1 al 30 que maneja el algoritmo mismo, los índices que se van seleccionando, y cómo estos se traducen en la selección de ciertas personas para los cargos de vocal de mesa titular y reemplazante.

Llegando a este punto, en donde se tiene la nómina con los candidatos y se conoce el procedimiento para elegir vocales de mesa, la herramienta faltante para poder reconstruir la selección independientemente es un generador de números aleatorios, el cual en la explicación del algoritmo de sorteo simplemente se hace el supuesto de que existe. En parte, esto se hace porque si se explicase la generación de números aleatorios al mismo tiempo que el sorteo, sería demasiada información dada de una vez, lo que podría resultar abrumante. Por otro lado, de esta forma se explica primero lo más concreto y menos técnico, para luego ir ahondando a medida que se avanza en la página. Así, la sección que sigue es sobre la generación de números pseudoaleatorios.

Generador Pseudoaleatorio y Semilla de Azar

En “Generador Pseudoaleatorio y Semillas de Azar”, primero se explica qué es un generador pseudoaleatorio, y qué es una semilla y su uso. En la figura 5 en Anexos se puede ver la primera parte de esta explicación.

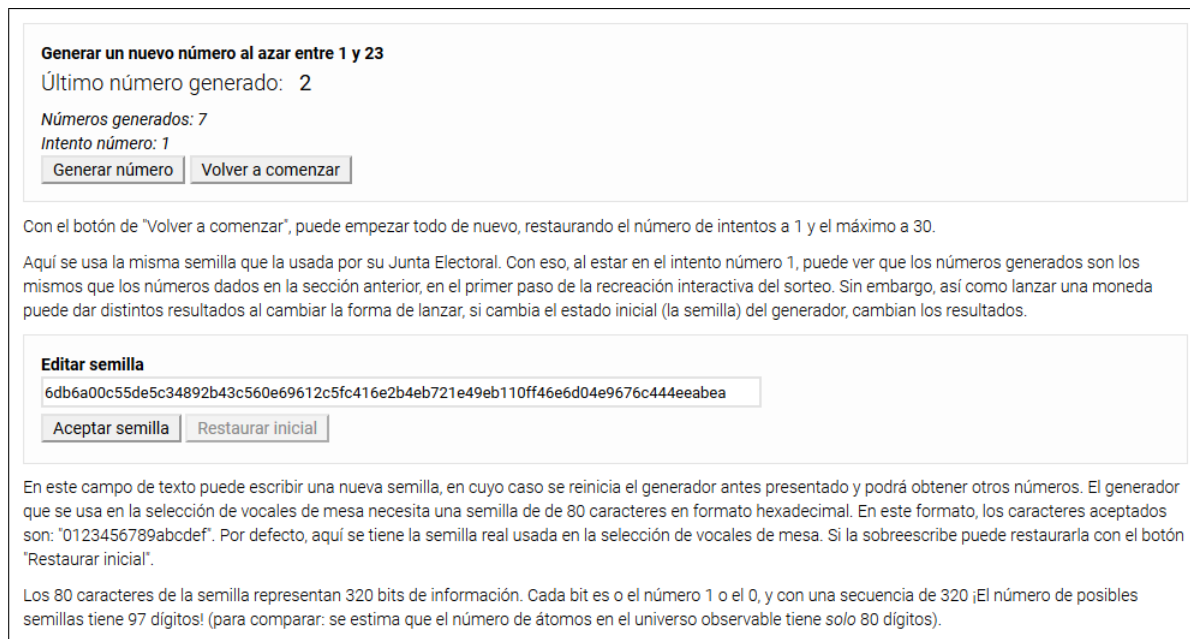


Figura 4.15: Captura de pantalla de explicación interactiva sobre generadores pseudoaleatorios y sus semillas.

Luego, viene una sección donde se presenta un generador pseudoaleatorio con el cual se puede interactuar para generar números, el cual es de hecho el mismo usado para la selección de vocales, con la misma semilla. En la figura 4.15 se observa el generador luego de haber generado 7 números (en la figura 6 en Anexos se muestra solo el generador en su estado inicial).

El generador da números entre los rangos que se usan en la selección de vocales, es decir, primero de 1 a 30, luego de 1 a 29, y así sucesivamente hasta haber generado 10 números. Luego de esto se reinicia el rango, pero aumenta el número de “intentos”, donde cada uno de estos intentos representa un sorteo realizado con ese generador, con la semilla dada.

Esta sección tiene también una segunda parte, donde se puede cambiar la semilla original. Si se sobrescribe la semilla que viene por defecto, se reinicia el estado del generador con esa semilla y se podrá ver que se obtienen otros números. En la misma figura 4.15 se puede ver el campo para cambiar la semilla.

Luego de las regiones interactivas, se comenta sobre conceptos de seguridad y se precisa el generador pseudoaleatorio usado (dando un vínculo al código). Se usan vínculos externos porque ese contenido se considera un asunto demasiado técnico para este contexto. Una captura de esta parte se muestra en la figura 7 del Anexo.

Con esto se provee no solo una herramienta que ayuda a entender la generación de números aleatorios y el uso de semillas, sino que también una pieza más de la verificación de la selección de vocales. En particular, se puede ver cómo se generan los mismos números que en la sección donde se explica el algoritmo de sorteo. Para ejemplificar esto, lo mostrado por la figura 4.15 corresponde a la iteración que se está desarrollando en la figura 4.14.

En esta sección, de nuevo por simplicidad, no se especifican detalles sobre cómo se obtiene la semilla (si no que solo se introduce al concepto de la misma. Esto se hace en la siguiente sección.

¿Cómo se obtiene la semilla de azar?

La siguiente sección va más a fondo en cómo se obtiene la semilla y, a su vez, se divide en tres subsecciones.

En la primera sección se habla sobre el faro y los pulsos de aleatoriedad del CLCERT. Se dan también recursos hacia los sitios del CLCERT relacionados.

La segunda sección trata sobre la verificación del pulso de aleatoriedad. Se introduce la idea de verificabilidad del pulso y se ofrecen recursos relacionados, como vínculos a una aplicación web que verifica pulsos y a la página para desarrolladores de Random UChile.

La tercera parte trata de cómo se procesa la aleatoriedad del pulso antes de usarla como semilla. Se habla de las funciones de hash y se explica el proceso mencionado en la sección 4.2.3, por medio del cual cada junta electoral obtiene una semilla (muy probablemente) distinta. Para acompañar el texto, se tiene una región interactiva que permite introducir un valor aleatorio en hexadecimal y un texto cualquiera al que se le llama “Identificador” para producir un hash. Esta región se puede ver en la figura 4.16.

Luego de la región interactiva se mencionan otros conceptos relacionados a las funciones de hash (como una idea general de nociones de seguridad para estas funciones) para terminar la sección. El texto de estas tres partes puede verse en la figura 9 de Anexos. El texto final recién mencionado se puede ver en la figura 10 también en los Anexos.

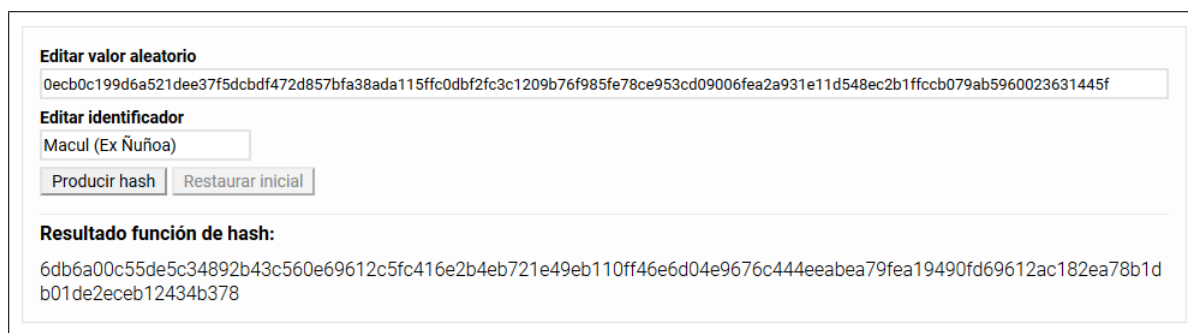


Figura 4.16: Captura de pantalla de explicación interactiva sobre funciones de hash.

Ejemplo Completo

Para concluir, se deja un ejemplo completo al final de la explicación avanzada, que permite explorar la combinación de los conceptos vistos a lo largo de todo. En la figura 10 en Anexos se muestran las palabras finales, que introducen a la última región interactiva.

En la figura 4.17 se muestra esta última región interactiva, que combina todos los conceptos vistos a lo largo de toda la explicación avanzada. Permite recrear el sorteo, haciendo reintentos y produciendo nuevas semillas de la misma forma como en realidad se hace para el sorteo verificable.

Las interfaces son muy similares a las previas explicaciones interactivas, pero con ligeros cambios para mostrar como los conceptos se integran. Se modifican algunos de los textos, el componente que produce semillas ahora marca cuál porción del hash de 512 bits se usa como semilla, y la recreación del sorteo tiene ahora una interfaz que demuestra el concepto de reintentos del sorteo.

El objetivo de esta última parte es simplemente ofrecer una herramienta para explorar a gusto todo lo antes descrito y ver cómo los conceptos se integran juntos. Cabe destacar que el concepto de “componentes” de Vue.js fue útil para poder combinar distintas regiones interactivas en una sola, cambiando partes de su interfaz según se necesitara.

4.7.4. Página de Detalles Técnicos del Sorteo

Como se ha mencionado en algunas secciones anteriores, se crea también una página con detalles técnicos del sorteo. Ésta está pensada como un resumen de un sorteo verificable a nivel general, y no para una mesa en específica; dirigida a gente que tenga suficiente conocimiento técnico como para entender más fácilmente los conceptos involucrados (por ejemplo, se incluye pseudocódigo). Esto en contraste a la páginas de las que se ha hablado previamente, que van dirigidas a explicar todo de forma más detallada y ejemplificada.

En la figura 4.18 se ve una porción de los descrito en esta página. En la figura 13 en los Anexos se ve una captura de la página completa.

Editar valor aleatorio
0ecb0c199d6a521dee37f5dcbdf472d857bfa38ada115ffc0dbf2fc3c1209b76f985fe78ce953cd09006fea2a931e11d548ec2b1ffccb079ab5960023631445f

Editar identificador
Macul (Ex Nuñoa)

Producir semilla Restaurar inicial

Semilla producida:
6db6a00c55de5c34892b43c560e69612c5fc416e2b4eb721e49eb110ff46e6d04e9676c444eeabea79fea19490fd69612ac182ea78b1db01de2eceb12434b378

Siguiente paso Saltar hasta selección

Intento número 1

Saltar a siguiente intento

Eligamos una posición al azar entre 1 y 27 (que es el largo de la lista de números disponibles).
Se escoge la posición: 19

Ver la lista, y escoger el número que esté en esa posición. En este caso se toma el número **19**.

Lista de números disponibles:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 23, 26, 27, 28, 29, 30.

Quitar ese número de nuestra lista, y escoger de vocal a la persona que esté representada por ese número.
Es decir, ahora se eligió a la persona número **19: M. S. Cristóbal M.**

Números sorteados:
25, 22, 24, 19.

Resultados

Vocales titulares:
25: R. A. Alejandro A.
22: P. C. Franco A.
24: R. S. Matías J.
19: M. S. Cristóbal M.

Reemplazantes:

Figura 4.17: Captura de pantalla del ejemplo interactivo completo.

Mientras que otras páginas de explicación mencionadas serían accedidas a través de identificar a quien las visite, por medio de su RUN, en este caso la página está parametrizada por el nombre de la junta electoral. El nombre de la junta electoral se escribe en la parte final de la ruta de la URL (para hacer más fácil escribir la ruta manualmente, al procesar el nombre no se distingue entre mayúsculas y minúsculas).

La información dada en la página, incluyendo vínculos a recursos externos como una implementación del generador pseudoaleatorio usado y la URI del pulso aleatorio, deberían permitir a un desarrollador poder replicar el sorteo.

4.7.5. Verificando la validez del Pulso

En las secciones anteriores, se ha mostrado que hay explicaciones y demostraciones, a distintos niveles, de cómo se lleva a cabo una selección verificable de vocales de mesa. Se dan aquellas explicaciones; se da la nómina de candidatos, el pulso de aleatoriedad, el generador pseudoaleatorio y los procedimientos que se siguen para obtener los números. Con ello, se

Detalles Técnicos - Sorteo Junta Electoral Macul (Ex Ñuñoa)

Escuchar

NÚMEROS SORTEADOS: 25, 22, 24, 19, 21, 4, 2, 1, 5, 29.

NÚMERO DE REINTENTOS: 0

ALGORITMO PARA RECREAR EL SORTEO: Pseudocódigo, donde "sorteados" es la lista final de números sorteados:

```

1 for i desde 0 a número de reintentos:
2   sorteados = Lista vacía
3   candidatos = Lista de números del 1 al 30, ordenados de menor a mayor
4   for i desde 30 a 21:
5     posicion_al_azar = número aleatorio entre 1 e i (inclusivo)
6     sorteado = candidatos[posicion_al_azar]
7     candidatos.remove[posicion_al_azar]
8     sorteados.agregar_al_final[sorteado]
9   end for
10 end for

```

(en el código mostrado se indexa las listas desde 1)

GENERADOR PSEUDOALEATORIO: Para elegir un número al azar dentro de un rango, para la línea 5 en el pseudocódigo mostrado arriba, se usa un generador pseudoaleatorio de números basado en ChaCha20. El código de una implementación en JavaScript del generador se encuentra público en internet. Con esta implementación usaría la función "randInt" usando 1 como argumento para el parámetro "min" e "i" como el argumento para "max". Para inicializar el generador se requiere de una semilla de 320 bits, que se deriva a partir del valor aleatorio dado por el "pulso de aleatoriedad" obtenido del faro de aleatoriedad de la Universidad de Chile y del nombre de la Junta Electoral que realiza el sorteo. Se detalla más información sobre la semilla en las siguientes secciones.

URI PULSO DE ALEATORIEDAD USADO: <https://beacon.clcert.cl/beacon/2.0/chain/4/pulse/540384>. Valor aleatorio dado por la propiedad "outputValue".

Figura 4.18: Captura de pantalla de página de resumen técnico de un sorteo verificable.

Verificación de Pulsos CLCERT

ID del Pulso a verificar

295668

Resultados

¡Pulso correcto! El pulso de aleatoriedad fue verificado correctamente.

Figura 4.19: Captura de pantalla una aplicación web que, dado un ID de pulso de aleatoriedad, permite verificar ese pulso.

les está dando a las personas las herramientas para recrear la selección de vocales, y se espera también que se les estaría dando lo suficiente para demostrarse a sí mismas si ésta fue realizada correctamente. Al proveer recursos para aprender sobre el faro de aleatoriedad además, se dan herramientas para realizar la verificación de la integridad del fundamento de todo el proceso: el pulso de aleatoriedad. Esto es, cerciorarse de que un pulso fue efectivamente generado correctamente por el faro de aleatoriedad del CLCERT.

Realizar la verificación del pulso se entiende como un paso que cualquier persona con

conocimiento técnico puede realizar, si tiene una forma de identificar el pulso que se desea verificar. Pero también se consideró importante proveer un ejemplo funcional de un programa que hiciera ese trabajo, por esto se creó una aplicación web en la que se puede escribir un identificador de pulso para obtener los resultados de su validación. En la página prototipo de “Explicación Avanzada” mostrada en las secciones anteriores, podría ser, por ejemplo, esta aplicación la que se vincula al momento de explicar el concepto de la verificación.

En la figura 4.19 se ve una captura de pantalla del ejemplo de página de verificación, que funciona por medio de los scripts de verificación provistos por el CLCERT [15] (más específicamente, lo que se encuentra en `chain_consistency_version2.py`), pero modificados de forma que puedan usarse como una función provista por un servidor, a través de un punto API que recibe una ID de pulso, corre los scripts y responde con los resultados de los mismos. El programa de verificación se llama usando la opción “`--only`” con el identificador del pulso de interés, pasando “`CLCERT`” a la opción “`--organization`” y activando la opción “`-a`” (para correr todos los tests). Cabe mencionar que el programa de verificación puede demorar de uno a un par de minutos, pero de todas formas no tanto como para dejar de ser práctico en una aplicación web.

La página se pensó como externa al SERVEL mismo pues, en parte porque su único trabajo es dar una respuesta de “válido/inválido” (y mostrar los errores en el segundo caso) y no explicar, se considera que es más significativo que un agente externo, que pueda mostrarse como una tercera parte imparcial, es quien implemente el verificador.

En la sección de Anexos se dejan otras capturas de pantalla de la página; en la figura 12 se muestra un pulso que no ha pasado la verificación junto a los errores dados por los scripts de verificación (incidentalmente, obtener este resultado permitió al CLCERT corregir un error que había en cómo publicaba los pulsos de aleatoriedad), y en la figura 11 se muestra cómo se ve la página cuando está esperando los resultados.

Teniendo esta página como ejemplo, se demuestra que es factible para terceras partes implementar no solo verificaciones privadas del proceso, sino que también aplicaciones web que pueden estar potencialmente abiertas al público general. El potencial de que existan agentes externos que puedan verificar el pulso de aleatoriedad es, en sí mismo, el último paso para completar el diseño del sistema de verificación, que completa y da validez a las páginas donde se explica el proceso de selección en niveles más generales (pues parte de la idea de ellas es que el valor aleatorio con el que inicia todo puede considerarse de confianza).

Capítulo 5

Validación

5.1. Conformación a las Especificaciones

Usando las especificaciones dadas en el Capítulo 3 (Descripción del Problema) como referencia, se puede ver que la solución cumple con la mayoría de las características buscadas.

Para la generación de números pseudoaleatorios, se utiliza ChaCha20, ampliamente considerado seguro, como base directa de la librería de funciones aleatorias implementada. Los números generados son también fácilmente reproducibles, gracias a que el generador funciona por medio de semillas que se dan explícitamente al momento de inicializarlo, además de que está hecho para recibir directamente los valores aleatorios del pulso. Finalmente, no solo se ha implementado con portabilidad a otros lenguajes en mente, sino que también ha sido efectivamente adaptada a Python durante su desarrollo (por Camilo Gómez, con una limitada participación del memorista).

La fuente de aleatoriedad verificable elegida se considera ideal para el contexto, no solo por la conveniencia de los scripts verificadores, sino que también por el hecho de pertenecer a una institución nacional, lo que se considera una ventaja en cuanto a llevar esto a una implementación real.

El sistema de sorteo cumple con lo planeado, con el cuidado que en él se ha implementado una funcionalidad de “reintentos”, a fin de ser más flexibles con las juntas electorales. Por un lado, esto les entrega a ellas la habilidad de probar otros números para el sorteo, lo que puede inducir a una manipulación de los resultados. Por otro lado, se ha implementado de tal manera que quede registro de cómo se usa esta funcionalidad, lo que transparenta las acciones de las juntas electorales, y da herramientas a las partes verificadoras para observar y fiscalizar su comportamiento. Es posible que en el futuro se cambie o remueva la habilidad de reintentar el sorteo.

Para la verificación, se logran dar prototipos de qué datos guardar y cómo, de forma que luego se pueda verificar. También se dan prototipos de cómo comunicar esta información al público, tanto a aquellas personas con conocimiento técnico como al público en general.

Se prueba además implementando una aplicación de verificación de pulsos (que fue detallada en la sección 4.7.5), la cual funciona correctamente. A ejemplo de esto último, gracias a que se observaron pulsos de aleatoriedad evaluados como erróneos por la aplicación, Random UChile pudo detectar un error en su sistema y arreglarlo.

5.2. Pruebas

Para demostrar los sistemas, se crean datos de prueba que permitieron simular a una junta electoral usando el sistema de sorteo verificable, y la asignación de los resultados de este sorteo a mesas receptoras específicas, creando así una lista de vocales de mesa (incluyendo el caso donde la mesa tenía vocales elegidos previamente, implicando que no todas las personas sorteadas terminaban siendo vocales de mesa). Llevar a cabo el sorteo exitosamente también implica que se validan los números que el frontend reporta al backend, lo que valida la reproducibilidad del algoritmo de generación pseudoaleatoria de números a través de distintos lenguajes (en este caso, se toma un sorteo realizado en JavaScript y se valida en Python). Se observó también que toda la información necesaria para la verificación estuviese registrada y disponible en el servidor.

Usando los datos obtenidos de simular sorteos exitosamente, se pasa a pruebas de las páginas de explicación. Se toma una muestra de tres personas cercanas al memorista, no relacionadas al proyecto. Con esta muestra se hacen pruebas informales de usuario de aquellas páginas para comunicar la información del sorteo a través del sitio web del SERVEL. Ninguno de los usuarios conocía el concepto de aleatoriedad verificable de antemano, todos son adultos en trabajos no relacionados a las ciencias de la computación, si bien uno de ellos tiene un conocimiento básico sobre HTML y páginas web. Dos han sido vocales de mesa en el pasado (uno múltiples veces).

Las pruebas de usuario consistieron en poner a los sujetos en dos escenarios, en los que se les pidió pensar en voz alta y se observó su comportamiento. En el primero, se les presenta la página modificada de Consulta SERVEL (donde se les ha elegido de vocales), con la premisa de que están viendo sus datos electorales y pueden navegar el sitio como deseen. El segundo escenario comienza en forma similar, pero se les sugiere ahora a no confiar en el sorteo que les habría elegido de vocales de mesa, y navegar el sitio bajo esa perspectiva. En ambos casos se les pide a los sujetos explorar las páginas a gusto, parando en el momento que deseen. Al final del proceso se les hacen las siguientes preguntas: si confían en el sorteo, y, si lo hacen, en qué punto se consolida esa confianza; si sienten alguna frustración no satisfecha al respecto de los prototipos; qué cambiarían o agregarían, y finalmente, se les pregunta por comentarios en general.

Estas pruebas fueron generalmente exitosas y sirvieron también para mostrar tres tipos distintos de visitantes de las páginas, con distintos niveles de curiosidad y confianza. En los siguientes tres párrafos se discuten los puntos más relevantes de cómo los distintos sujetos han respondido a las preguntas hechas y cómo actuaron en los dos escenarios presentados.

El primer sujeto reportó que el certificado, la sección interactiva de “Recreación del Sor-

teo” y lo que leyó en el vínculo al sitio “Acerca de” de Random UChile fueron suficientes para consolidar su confianza, remarcando que las secciones interactivas y lo que se explica en la página de Random UChile le fueron muy interesantes. El segundo sujeto revisó el certificado e hizo una lectura rápida y superficial de la página de explicación detallada, reportando que los conceptos que observó involucrados, lo que aprendió de lo que vio, y la aparente envergadura y complejidad del proceso de la aleatoriedad verificable fueron lo que consolidaron su confianza; el segundo sujeto además no se molestó en visitar la página de detalle técnico del sorteo verificable, pues dedujo (correctamente) de lo dicho en Consulta SERVEL que no era su público objetivo. El tercer sujeto se muestra más curioso durante el segundo escenario, dándose el tiempo de visitar cada página relacionada; este sujeto se muestra especialmente escéptico y revisa la página de explicación detallada primero superficial y rápidamente, luego volviendo atrás para aclarar las dudas que le iban surgiendo, interactuando con cada sección interactiva y vínculo; inicialmente se muestra desconfiado por la complejidad del asunto, pero va consolidando su confianza al leer la explicación e interactuar con la página. El tercer sujeto finalmente reporta estar satisfecho, diciendo que su confianza se consolida gracias a la completitud del detalle ofrecido, y especialmente gracias a las secciones interactivas que le permiten explorar los conceptos.

Todos los sujetos reportaron sentirse satisfechos con las páginas y no quedar con frustraciones luego de navegarlas. Durante el primer escenario, todos expresan apreciación por la nueva información añadida a Consulta SERVEL y por la primera sección de la página de explicación detallada, pues no conocían el procedimiento seguido para elegir vocales de mesa. Se observó que todos los usuarios se mostraron interesados en los conceptos relacionados a aleatoriedad verificable durante las pruebas; efectivamente, al final de la prueba, durante la sección de preguntas, todos reportan explícitamente que así les pareció. Dos se muestran especialmente interesados no solo en la aleatoriedad verificable, sino que también en los conceptos de computación relacionados que se explican, en especial en la idea del faro de aleatoriedad, incluyendo lo que se describe en el sitio web de Random UChile, en la página “Acerca de”.

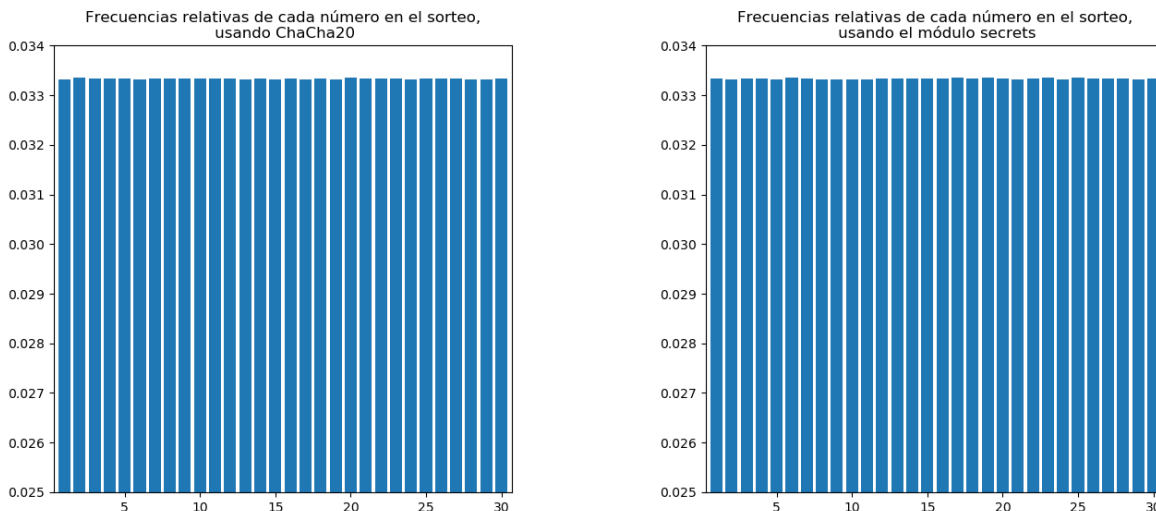
De las pruebas se obtuvo retroalimentación útil que permitió realizar pequeños pero notables cambios a los prototipos, para mejorar la experiencia de usuario. Entre los comentarios estaban: una sugerencia de añadir información sobre el bono que, por ley, se les debe conceder a los vocales de mesa que cumplan con su rol; enfatizar que los campos de texto de las secciones interactivas son editables (derivo en que su etiqueta ahora comience con “Editar”), y algunas sugerencias sobre la redacción. Todos los cambios sugeridos se consideraron acertados y se integran de alguna forma a los prototipos.

En términos de la pseudoaleatoriedad misma del sorteo y de sus cualidades estadísticas, no se observan problemas en la generación de números a lo largo de todas las veces que se ha usado o se han realizan pruebas de cualquier tipo del sorteo. Se depende de la correctitud de la implementación de ChaCha20 que se usa, así como del entendimiento de que ChaCha20 es seguro que la comunidad criptográfica ha ganado a lo largo del tiempo. Adicionalmente, a modo de ilustración y para asegurar que no hubiese una falla crítica u obvia en la generación de números (por ejemplo: que no hubiese un error en el cual uno de los números nunca fuese seleccionado), se simula múltiples veces solo la porción de selección de números del sorteo, para observar las frecuencias de las apariciones de los distintos números en los resultados.

Se simularon 300 mil millones de sorteos para dos distintos generadores pseudoaleatorios (es decir, 600 mil millones de sorteos en total, cada uno consistiendo en obtener 10 números al azar, sin repetición, a partir de una lista de números del 1 al 30).

En el primer caso, se usa ChaCha20 como en el resto del trabajo de memoria, en el segundo se usa la función `randbelow(n)` del módulo `secrets` de Python, que (según la documentación del mismo) sirve para generar números aleatorios criptográficamente fuertes, por lo que se consideró que sirve como una buena referencia. Para cada una de las 300 mil millones de simulaciones con ChaCha20 se usa una semilla distinta, lo que se logra generándolas a partir de SHA-3 512, alimentando a la función de hash con una representación en bytes del índice de cada simulación (indexadas desde 0); el módulo `secrets`, por su parte, obtiene sus semillas automáticamente.

Con esto se exploraron dos propiedades: primero, las apariciones de cada número del 1 al 30 (inclusivo) en los resultados del sorteo, lo que se ve en la figura 5.1 donde se usan frecuencias relativas. Segundo, se ven las apariciones de cada número en cada posición en la que pueden aparecer en los resultados del sorteo (como se sortean 10 números, hay 10 posibles posiciones en las que un número puede aparecer). En la figura 5.2 se ven las frecuencias relativas con las que cada número del 1 al 30 fue el primer número seleccionado por las simulaciones de sorteo. Lo que se espera es que los resultados entre ambos generadores sean muy parecidos, así como que las frecuencias relativas entre ellas también sean todas extremadamente similares, reflejando un comportamiento de distribución uniforme.



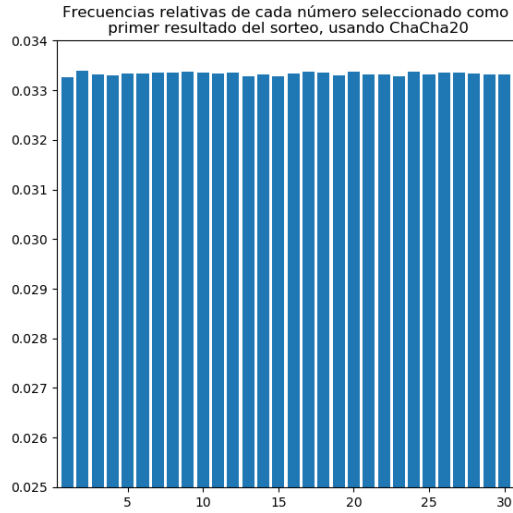
(a) Frecuencias relativas al usar ChaCha20.
 $\sigma = 1,0531377033523408e - 05$

(b) Frecuencias relativas al usar `secrets`.
 $\sigma = 9,971812049751523e - 06$

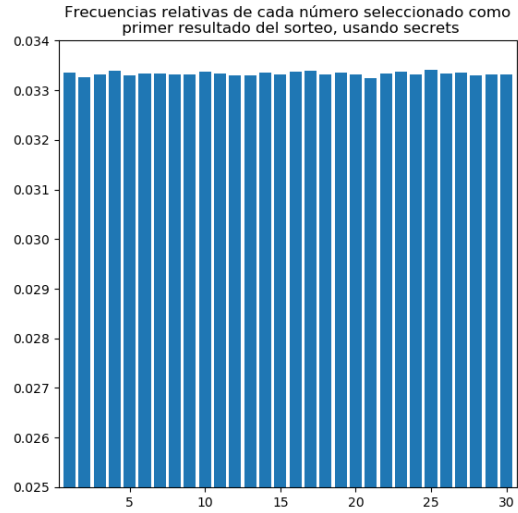
Figura 5.1: Comparación de las frecuencias relativas de números sorteados, luego de simular 300 mil millones de sorteos con dos distintos generadores.

En el caso de la figura 5.1 se puede ver que para ambos casos las frecuencias relativas son muy parecidas, y no parece haber ningún problema obvio o crítico con ChaCha20.

En la figura 5.2 de nuevo se observa que para ambos casos las frecuencias relativas son



(a) Frecuencias relativas al usar ChaCha20.
 $\sigma = 3,158962558253061e - 05$



(b) Frecuencias relativas al usar `secrets`.
 $\sigma = 3,5206932187767754e - 05$

Figura 5.2: Comparación de las frecuencias relativas para el primer número elegido durante los sorteos, luego de simular 300 mil millones de sorteos con dos distintos generadores en cada caso.

muy parecidas y tampoco parece haber ningún problema obvio o crítico. Estas pruebas se realizaron para todas las 10 posibles posiciones en las que un valor puede estar al término de un sorteo, pero en cada caso lo que se observa es muy similar a lo mostrado en la figura 5.2.

En cualquier caso, se hace hincapié en que estas simulaciones son solo para ilustrar y descartar fallos críticos en el uso que se le ha dado a ChaCha20 en este contexto, dando seguridad a que el generador se ha usado correctamente. La seguridad de ChaCha20 en sí se desprende de que éste ya ha pasado por el escrutinio de la comunidad criptográfica, llevándolo a ser entendido como un generador pseudoaleatorio seguro, hecho que da soporte crucial a la seguridad de la generación de números durante los sorteos.

Capítulo 6

Conclusión

6.1. Resumen

En este trabajo de memoria se ha creado un prototipo de un sistema que demuestra una implementación del uso de aleatoriedad verificable en el sorteo de vocales de mesa.

Se implementa una librería de funciones pseudoaleatorias basadas en ChaCha20, en JavaScript. Los resultados de estas funciones son fácilmente reproducibles (por depender de una semilla), es fácil utilizarla con el valor de un pulso de aleatoriedad, y puede ser (y ha sido) adaptada a otros lenguajes.

Con el propósito de implementar el sorteo verificable en el ambiente real, se identifican los datos que el SIVOME debe poder acceder y manejar, así como procesos que deben existir en el servidor para la implementación en producción.

Se diseñan los flujos de información entre las distintas partes, incluyendo restricciones que deben imponerse para asegurar la correctitud del proceso.

También se diseña una manera de explicar el procedimiento a aquellas personas que sean sorteadas. Ésta incluye formas interactivas de explorar los conceptos relacionados, y empieza desde las ideas más simples y generales, para luego ir avanzando, paulatinamente, desde ellas hasta las más complejas y técnicas.

Se hacen al final pruebas de usuario con tres personas cercanas al memorista, que resultan exitosas y productivas.

6.2. Revisión de Objetivos

Con respecto a los objetivos (descritos en la sección 1.2): en general se alcanzan los objetivos relativos a diseñar procesos, conexiones entre sistemas, e interfaces. En lo siguiente la

numeración dada en esa sección.

Los objetivos 1 y 2 son alcanzados por lo descrito en la secciones 4.5 (Backend) y 4.6.2 (Nuevo Sorteo). En específico, esto se logra diseñando el manejo de datos relacionados al sorteo verificable en un backend, diseñando la obtención de pulsos de aleatoriedad, y finalmente creando una interfaz prototipo para el SIVOME, que permitiría a las juntas electorales realizar el sorteo verificable.

Para el objetivo 2, si bien la interfaz es diseñada para que le sea amigable a las juntas electorales, falta aún seguir refinando y probando el prototipo con retroalimentación de las juntas electorales.

Para el objetivo 3, se diseñan prototipos de interfaces que permitirían dar a vocales de mesa sorteados la oportunidad de verificar su selección, en distintos niveles. Como se muestra en la sección 4.7, primero se les ofrece simplemente un certificado de aleatoriedad de los entregados por el CLCERT. Si eligen indagar más, los vocales seleccionados por sorteo verificable pueden ir aprendiendo sobre cómo funciona este sistema. Con ello se dan herramientas que permiten recrear el sorteo de vocales de mesa y verificar sus resultados.

Queda también, en el futuro, refinar lo hecho para el objetivo 3, en especial probar y mejorar las explicaciones dadas que, por la complejidad técnica de los conceptos involucrados, son difíciles de hacer simultáneamente simples y completas. La manera de abordar dicha complejidad en esta memoria fue introducir los conceptos estimados como más técnicos en forma gradual, partiendo desde lo más general. Al mismo tiempo que se crea una explicación alternativa, dirigida a quienes tengan una base técnica.

6.3. Discusión de Resultados

Un punto de esta memoria en cual queda trabajo por hacer, es el relativo a prueba formales con usuarios reales. Si bien durante el trabajo los sistemas desarrollados pasaron por varias iteraciones, y se testeó parte del sistema de verificación con sujetos cercanos al memorista, el alcance de estas pruebas fue limitado.

No obstante, de las pruebas con usuarios, se destaca lo positivo del resultado, en especial que los usuarios apreciaron la completitud de las explicaciones y recursos dados. Los sujetos de prueba quedan satisfechos y sienten haber aprendido algo de su experiencia, si bien todos los revisaron a distintos niveles de profundidad.

Se destaca también que todos los sujetos de prueba expresan que el concepto de aleatoriedad verificable les parece novedoso e interesante. Además, los sujetos expresan haber aprendido sobre el proceso de sorteo en general, incluso siendo que dos sujetos han sido vocales de mesa. Expresan también lo útil de tener la cartilla de instrucciones disponible en la página de Consulta SERVEL. Todo esto es prueba de lo útil de añadir más información a la de Consulta de Datos Electorales.

Los objetivos sí fueron alcanzados, y se abordaron todos los problemas especificados. Se

destacan el diseño de interacción entre sistemas, el énfasis en llegar a personas que pueden no tener conocimiento técnico y la implementación de librería de funciones aleatorias.

Con respecto al impacto de este trabajo de memoria: considerando que no se implementó en el ambiente real, el impacto inmediato de esta memoria es limitado, pero de todas formas sirve para sentar bases y como estudio del problema.

La librería de funciones aleatorias se obtiene como un producto derivado de esta memoria, que ya puede ser usado para otros propósitos. Cabe mencionar además que la manera en la que se usan estas funciones puede verse como un proceso bastante genérico, que puede imitarse para otros casos de uso de aleatoriedad verificable, no solo en el uso directo de las funciones pseudoaleatorias, sino que también incluyendo el uso de una función de hash para individualizar la semilla y la propuesta de usar un estándar para ordenar alfabéticamente los identificadores de los objetos o personas a sortear. Así mismo, si bien las páginas que explican la selección de vocales con secciones interactivas se han desarrollado bastante a la medida de este caso de uso en específico, también puede ser útil tomar elementos de lo que se hizo ahí y modificarlos según sea adecuado para explicar los procesos de otros escenarios de uso del faro de aleatoriedad; por ejemplo, se pueden reproducir las secciones interactivas y ajustar sus parámetros y el texto de las explicaciones según sea el caso.

Esta memoria aborda un problema interesante para la aleatoriedad verificable, en el cual la parte usuaria de la aleatoriedad está dividida en dos. En este caso, esas dos partes son las juntas electorales y SERVEL, las primeras usando el software implementado por la segunda. La parte verificadora puede no confiar en una o en ambas de esas entidades, o si quiera entender que son separadas. Así, el trabajo aquí realizado puede servir para afrontar mejor escenarios como éste en el futuro.

Se espera que el trabajo hecho pueda ser un paso en la actualización de cómo se realiza el sorteo de vocales de mesa durante procesos electorarios chilenos. Para ir aún más lejos, se espera que una implementación en producción potencie la difusión del concepto de aleatoriedad verificable, fomentando la replicación de iniciativas como la de esta memoria.

6.4. Trabajos Futuros

Claramente, el trabajo futuro principal es llegar a una implementación de sorteo verificable en el ambiente real. También al hacer eso, se puede aprovechar de agregar otra información útil a la página de Consulta SERVEL. El trabajo a futuro que se espera realizar requiere, en especial, una colaboración más cercana con las juntas electorales y el SERVEL.

En el futuro, es importante también asegurar la accesibilidad (en el sentido del uso de lectores de pantalla, y similar) de las interfaces, pues si bien se hicieron los prototipos tratando de seguir prácticas de diseño relacionadas, esta propiedad no se han testeado. También es importante considerar proveer la información en otros lenguajes.

Cabe mencionar que en Agosto del 2019 (cerca del final del periodo dado a este trabajo de memoria), la página de Consulta de Datos Electorales del SERVEL sufrió una actualiza-

ción. Desde el punto de vista de usuario es más que nada estética y diseño más adaptable (*responsive*), pero en términos de implementación se usa ahora Vue.js, lo que acerca más el prototipo al ambiente real.

Este trabajo de memoria motiva a la pregunta de si se podrían implementar servicios web de verificación masiva de los sorteos de vocales en masa, en lugar de solo un sorteo asociado a un RUN a la vez. Tal opción permitiría obtener estadísticas a nivel nacional de la designación de vocales. Sin embargo, se debe notar que porciones de los datos pueden considerarse personales y privadas, lo cual implica que implementaciones de esto deben considerar la privacidad de las personas. En este sentido, parte de la tarea a futuro sería estudiar maneras de permitir la verificación al mismo tiempo que se protejan todos los datos personales asociados, o estudiar más a fondo los balances entre transparencia y privacidad en este sistema.

Bibliografía

- [1] “Ley 18700: LEY ORGANICA CONSTITUCIONAL SOBRE VOTACIONES POPULARES Y ESCRUTINIOS.” Diario Oficial de la República de Chile, Santiago, Chile, 06 de Mayo 1988. (Tiene Texto Refundido: Decreto con Fuerza de Ley N° 2-2017-09-06).
- [2] CLCERT, “Acerca de - RandomUChile.” <https://web.archive.org/web/20190622233745/https://random.uchile.cl/about/>. Consultado: 2019-06-22.
- [3] Franco Pino Córdova, “ChaCha20-Generator-Utilities.” <https://github.com/FrancoPinoC/ChaCha20-Generator-Utilities/tree/master/js>.
- [4] Camilo Gómez, “ChaCha20-Generator-Utilities - Python.” <https://github.com/FrancoPinoC/ChaCha20-Generator-Utilities/tree/master/python>.
- [5] SERVEL, “Más de 42 mil mesas receptoras de sufragios funcionarán en estas elecciones,” *Noticias Electorales*, 2017. <http://web.archive.org/web/20190819035550/https://www.servel.cl/mas-de-42-mil-mesas-receptoras-de-sufragios-funcionaran-en-las-elecciones-2017/>, Consultado: 2019-08-18.
- [6] W. Robbins, “2 GUILTY OF BID TO RIG PENNSYLVANIA LOTTERY,” *The New York Times*, p. 20, Mayo 1981.
- [7] International Organization for Standardization, “ISO/IEC 14651:2019 - Information technology – International string ordering and comparison – Method for comparing character strings and description of the common template tailorable ordering,” Enero 2019.
- [8] The Unicode Consortium, “Unicode Technical Standard #10: Unicode Collation Algorithm, Version 12.1.0,” Mayo 2019.
- [9] “International Components for Unicode.” <https://web.archive.org/web/20190730030856/http://site.icu-project.org/home>. Consultado: 2019-07-30.
- [10] “Consulta de Datos Electorales.” <https://web.archive.org/web/20190623001358/https://consulta.servel.cl/>. Consultado: 2019-06-22.
- [11] NIST, Computer Security Resource Center, “Interoperable Randomness Beacons.” <https://web.archive.org/web/20190629175513/https://csrc.nist.gov/projects/interoperable-randomness-beacons>. Consultado: 2019-06-30.

- [12] Inmetro, Divisão de Metrologia em Telecomunicações, “Inmetro’s Randomness Beacon.” <https://web.archive.org/web/20190827052105/https://beacon.inmetro.gov.br/records>. Consultado: 2019-08-27.
- [13] Dina Kozlov (Cloudflare), “League of Entropy: Not All Heroes Wear Capes.” <https://web.archive.org/web/20190827053539/https://blog.cloudflare.com/league-of-entropy/>. Consultado: 2019-08-27.
- [14] “Development - Random UChile.” <https://web.archive.org/web/20190623005641/https://random.uchile.cl/development/>. Consultado: 2019-06-22.
- [15] Camilo Gómez, “CLCERT/NIST Randomness Beacon Verifier Scripts.” <https://web.archive.org/web/20190622052651/https://github.com/clcert/beacon-verifier>. Consultado: 2019-06-22.
- [16] “Documentación de random.seed en Python 2.” <https://web.archive.org/web/20190625212707/https://docs.python.org/2/library/random.html>. Consultado: 2019-06-25.
- [17] D. E. Knuth, *The art of computer programming*, vol. 2, ch. 3 *Random Numbers*, Section 3.4.2 *Random Sampling and Shuffling*. Reading, Mass: Addison-Wesley Pub. Co, 3rd ed., 1973.
- [18] “Documentación de rand en C++.” <https://web.archive.org/web/20190625223056/http://www.cplusplus.com/reference/cstdlib/rand/>. Consultado 2019-06-25.
- [19] R. McEvoy, J. Curran, P. Cotter, and C. Murphy, “Fortuna: Cryptographically Secure Pseudo-Random Number Generation In Software And Hardware,” *IET Irish Signals and Systems Conference*, 2006.
- [20] D. J Bernstein, “ChaCha, a variant of Salsa20,” *Workshop Record of SASC 2008: The State of the Art of Stream Ciphers*, Marzo 2008.
- [21] R. Kneusel, *Random numbers and computers*. Cham, Switzerland: Springer, 2018.
- [22] “Página Web de Vota Inteligente.” <https://web.archive.org/web/20190622040022/https://votainteligente.cl/>. Consultado: 2019-06-21.
- [23] “Descripción de Chile Transparente en la página de Vota Inteligente.” <https://web.archive.org/web/20190622042039/https://votainteligente.cl/organizacion/ChileTransparente>. Consultado: 2019-06-21. La página de Chile Transparente no pudo ser archivada, en la fecha de consultado esta es: www.chiletransparente.cl.
- [24] L. F. Sandra Wachter, Brent Mittelstadt, “Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation,” *International Data Privacy Law*, vol. 7, iss 2, pp. 76–99, Mayo 2017.
- [25] NIST, Computer Security Resource Center, “NIST Policy on Hash Functions.”

<https://web.archive.org/web/20190809182044/https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions>. Consultado: 2019-08-9.

- [26] “Documentación de la implementación de ChaCha20 en PyCryptodome.” <https://web.archive.org/web/20190626022551/https://pycryptodome.readthedocs.io/en/latest/src/cipher/chacha20.html>. Consultado: 2019-06-25.
- [27] Christopher Wellons, “chacha-js.” <https://web.archive.org/web/20190626023102/https://github.com/skeeto/chacha-js>. Consultado: 2019-06-25.
- [28] Daniel Lemire, “Picking distinct numbers at random: benchmarking a brilliant algorithm (JavaScript edition).” <http://web.archive.org/web/20190701013124/https://lemire.me/blog/2018/01/31/picking-distinct-numbers-at-random-benchmarking-a-brilliant-algorithm-javascript-edition/>. Consultado: 2019-06-30.

Anexos

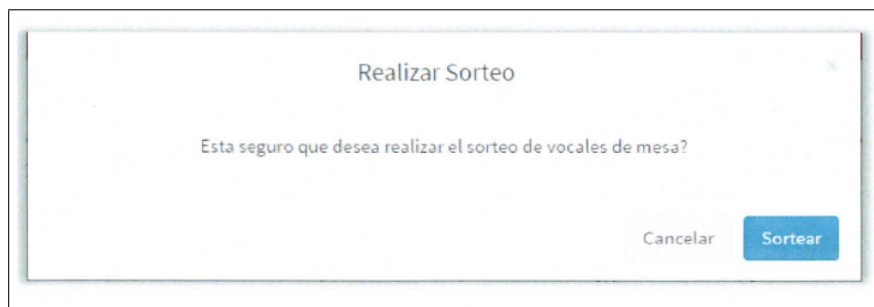


Figura 1: Imagen del modal para confirmar los números del sorteo, a selección, obtenida del manual del SIVOME.

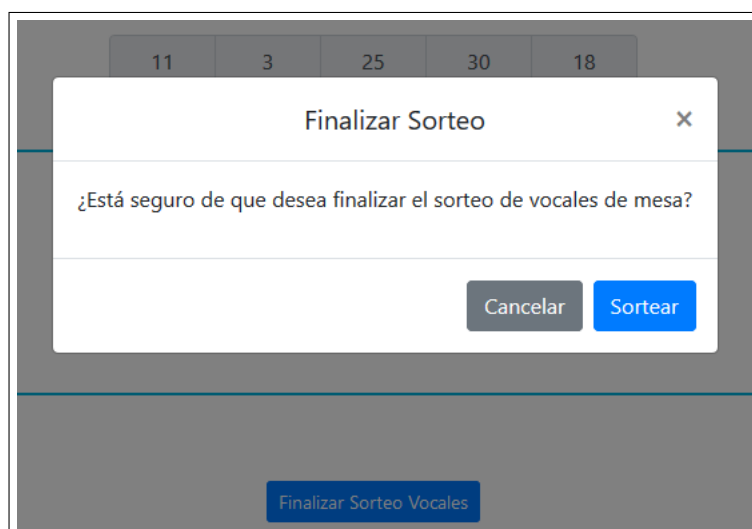


Figura 2: Modal para confirmar la finalización de la selección en el prototipo para página de sorteo verificable.



Figura 3: Captura de pantalla de la página que muestra el detalle de la selección de una persona como vocal de mesa. Se muestra como se marca una lista para indicar que se puede hacer clic en ella, con lo que luego se mostraría el siguiente paso.

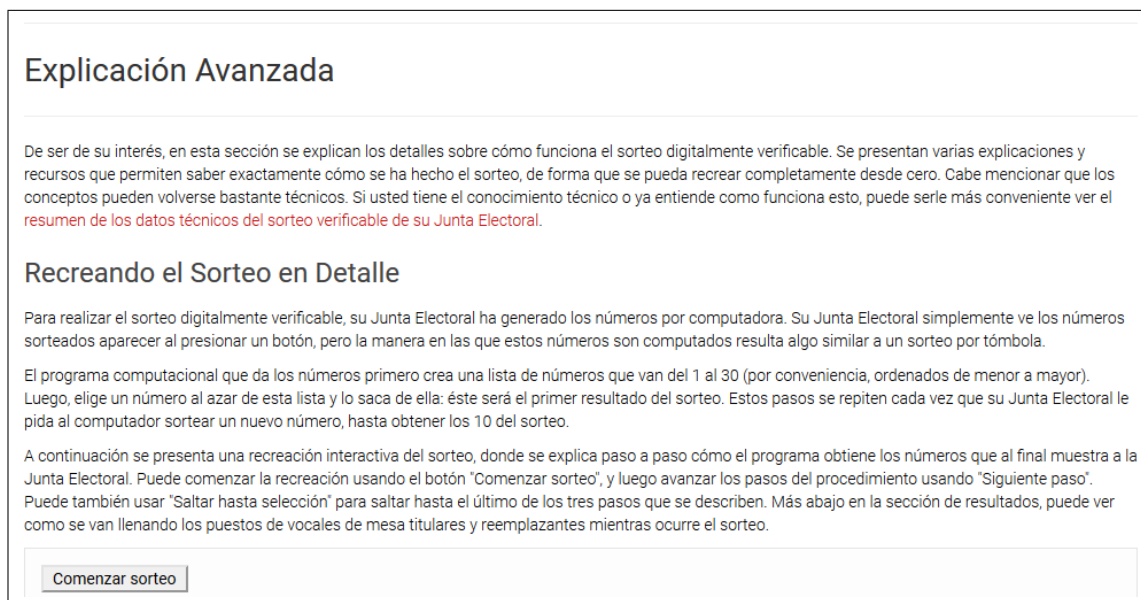


Figura 4: Captura de pantalla de la página que muestra el detalle de la selección de una persona como vocal de mesa. Aquí comienza la sección de explicación avanzada, que da detalles más técnicos. En la captura se ve un preámbulo a una sección interactiva mostrada en la sección 4.7.3, donde se puede ver cómo se eligen los 10 índices a partir de la semilla usada.

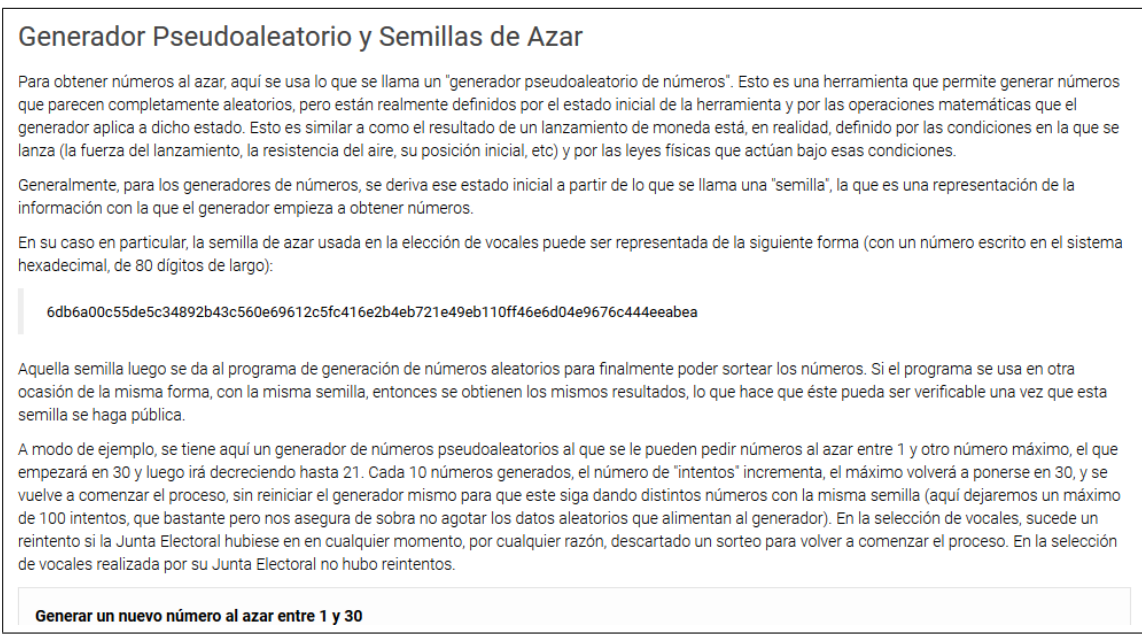


Figura 5: Captura de pantalla de la página que muestra el detalle de la selección de una persona como vocal de mesa. Aquí comienza la sección en donde se da una explicación sobre generadores pseudoaleatorios.

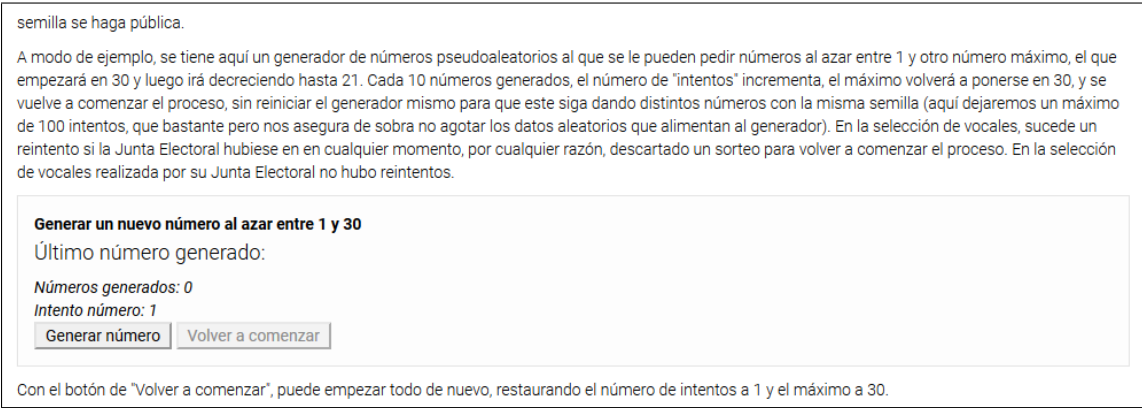


Figura 6: Captura de pantalla del generador pseudoaleatorio interactivo, en su estado inicial.

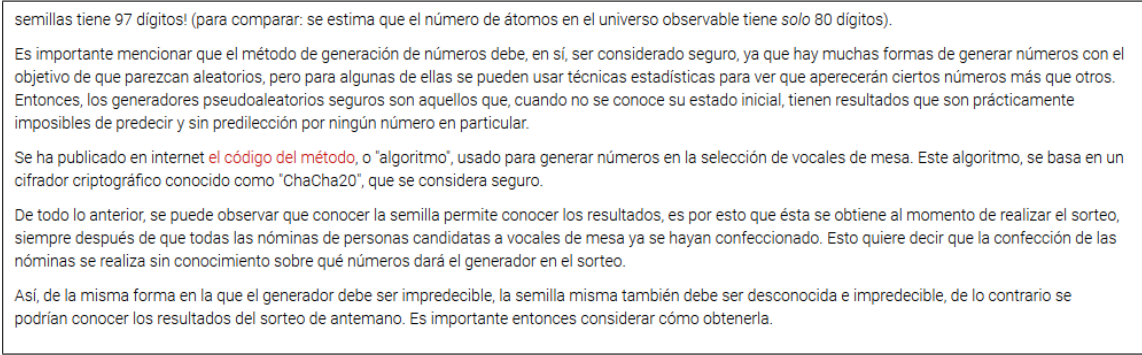


Figura 7: Captura de pantalla de la última parte de de la sección Generador Pseudoaleatorio y Semillas de Azar.

¿Cómo se obtiene la semilla de azar?

Faro de Aleatoriedad

La información aleatoria que se usa para crear la semilla de la selección de vocales es dada por la Universidad de Chile, a través del "Faro de Aleatoriedad" del Laboratorio de Seguridad Computacional y Criptografía Aplicada (CLCERT). Esta información impredecible se obtiene de varias fuentes, por ejemplo, desde datos de movimientos sísmicos en Chile, procesos cuánticos o del audio de la radio de la Universidad, entre otras.

Los datos dados por las fuentes se recolectan y combinan para formar lo que el CLCERT llama "pulso de aleatoriedad": un conjunto de datos que representa información aleatoria y las propiedades relativas al proceso de su generación para su posterior verificación. El Faro de Aleatoriedad publica uno de estos pulsos al inicio de cada minuto del día.

Para el sorteo de vocales de mesa, que debe realizarse a las 14:00 horas del trigésimo día anterior a la fecha de la elección o plebiscito que se vaya a llevar a cabo, se obtiene el primer pulso sin errores publicado desde las 13:59 horas en adelante. La información aleatoria dada por el pulso es entonces usada para derivar la semilla antes mencionada.

De ser de interés, la explicación más a fondo sobre la recolección de información aleatoria y por qué es de confianza puede encontrarse en el [sitio web del CLCERT](#).

Los datos técnicos del pulso desde donde se obtuvo la semilla mostrada pueden encontrarse en la página del [pulso de aleatoriedad](#). El campo "outputValue" indica el valor aleatorio asociado al pulso.

Verificación del Pulso de Aleatoriedad

El Faro de Aleatoriedad del CLCERT tiene como objetivo lograr aleatoriedad verificable, parte importante de lo cual es que los pulsos de aleatoriedad se construyan de tal manera que, si hubiese un error o una manipulación de los mismos, entonces estos problemas puedan detectarse con un proceso de verificación. Con ello, se puede asegurar que aquellos datos que son el fundamento de todo el proceso de selección son efectivamente válidos e impredecibles.

Cada pulso se identifica por medio de un número, en el caso de esta elección ese número es 540384. Sabiendo este número se puede entonces encontrar el pulso y verificarlo, para lo cual puede usarse una [aplicación web de verificación de pulsos aleatorios](#).

Si le interesa conocer más sobre el proceso de verificación y los pulsos de aleatoriedad, o si está interesado en implementar su propio programa de verificación, puede dirigirse a la [sección para desarrolladores](#) de la página del CLCERT.

Individualización de la Semilla

Dado que el mismo pulso de aleatoriedad se usa para todas las Juntas Electorales que usen el sistema de sorteo digitalmente verificable, se sigue un proceso para individualizar la semilla a cada Junta Electoral. De no hacer esto, los resultados de los sorteos de las Juntas Electorales serían todos iguales.

Figura 8: Captura de pantalla de una explicación sobre el origen de la semilla usada en el sorteo.

Individualización de la Semilla

Dado que el mismo pulso de aleatoriedad se usa para todas las Juntas Electorales que usen el sistema de sorteo digitalmente verificable, se sigue un proceso para individualizar la semilla a cada Junta Electoral. De no hacer esto, los resultados de los sorteos de las Juntas Electorales serían todos iguales.

Para lograr esto, se usa una función de "hash". De forma simplificada, ésta es un procedimiento matemático que recibe una cantidad cualquiera de datos y los transforma en una secuencia nueva de datos de un cierto largo fijo (a la que le llamamos el "hash" de los datos dados). Dicho de otro modo, el propósito de estas funciones es tomar toda la información que se les dé y mezclarla de forma azarosa, para producir una nueva pieza de información de un cierto tamaño. Esa pieza de información será lo que se usa al final como semilla para el generador pseudoaleatorio del sorteo.

Más concretamente, lo que se usa, para el sorteo de vocales de mesa, es la función conocida como "SHA-3 512". Una función de hash estándar en el mundo de la criptografía. Ella tomará secuencias de datos y dará una nueva secuencia de 512 bits como resultado.

Con esta herramienta, podemos individualizar la semilla a cada Junta Electoral de la siguiente forma:

1. Tomamos el valor del pulso de aleatoriedad dado por el CLCERT y lo convertimos en una secuencia de números entre 0 y 255, o bytes.
2. Luego, tomamos el nombre de la Junta Electoral, lo convertimos también en una secuencia de bytes, y la juntamos con la del pulso, para obtener una única secuencia. (la transformación de caracteres a bytes se realiza usando la codificación "UTF-8")
3. Finalmente, le damos esto a la función de hash y obtenemos 512 bits de información, de los cuales usamos los primeros 320 como semilla para el generador pseudoaleatorio.

En la siguiente demostración, se ve exactamente qué valores llegan a la función y qué resulta. Puede también cambiar los valores. El primero, que llamamos "valor aleatorio" es el valor del pulso de aleatoriedad y es lo que hace que el resultado de la función de hash sea impredecible y aleatorio, lo representamos en su formato en hexadecimal. El segundo, el "identificador", es un valor distinto para cada Junta Electoral, que hace que el resultado final sea (con muy extrema probabilidad) distinto para cada Junta Electoral. Si desea, puede probar cambiando los valores para ver qué sucede.

[Editar valor aleatorio](#)

Figura 9: Captura de pantalla de la página de explicación en detalle, donde se habla del proceso de hashing que produce la semilla.

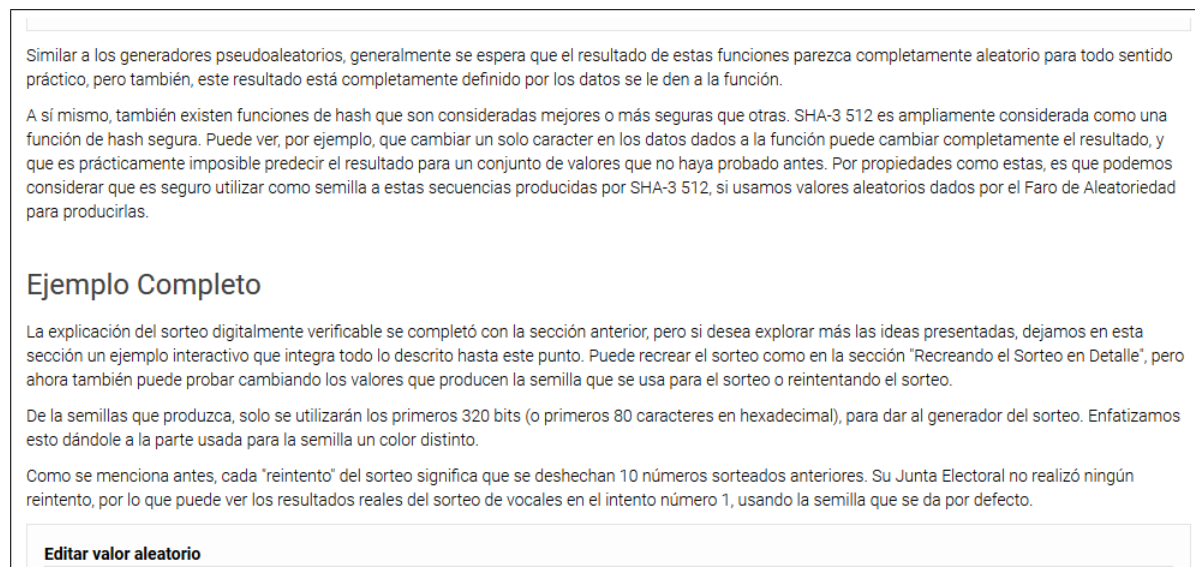


Figura 10: Captura de pantalla de la última sección de la página de explicación en detalle, donde se introduce el ejemplo completo de reconstrucción del sorteo (con texto de la sección de Individualización de la Semilla en la parte superior).



Figura 11: Captura de pantalla una aplicación web verificadora cargando información de un pulso.

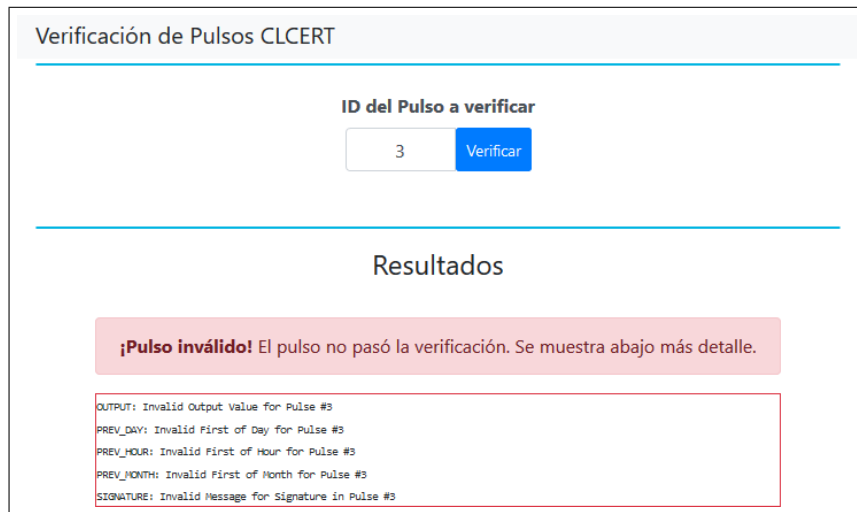


Figura 12: Captura de pantalla una aplicación web verificadora, donde el pulso falló la verificación.

VEL
SERVEL INFORMACIÓN ELECTORAL ESTADÍSTICAS PARTIDOS TRÁMITES PREGUNTAS FRECUENTES SALA DE PRENSA CONTACTO

Detalles Técnicos - Sorteo Junta Electoral Macul (Ex Ñuñoa)

Escuchar

NÚMEROS SORTEADOS: 25, 22, 24, 19, 21, 4, 2, 1, 5, 29.

NÚMERO DE REINTENTOS: 0

ALGORITMO PARA RECREAR EL SORTEO: Pseudocódigo, donde "sorteados" es la lista final de números sorteados:

```

1 for i desde 0 a número de reintentos:
2   sorteados = Lista vacía
3   candidatos = Lista de números del 1 al 30, ordenados de menor a mayor
4   for i desde 30 a 21:
5     posicion_al_azar = número aleatorio entre 1 e i (inclusivo)
6     sorteado = candidatos[posicion_al_azar]
7     candidatos.remove[posicion_al_azar]
8     sorteados.agregar_al_final[sorteado]
9   end for
10 end for

```

(en el código mostrado se indexa las listas desde 1)

GENERADOR PSEUDOALEATORIO: Para elegir un número al azar dentro de un rango, para la línea 5 en el pseudocódigo mostrado arriba, se usa un generador pseudoaleatorio de números basado en ChaCha20. El código de una implementación en JavaScript del generador se encuentra público en internet. Con esta implementación usaría la función "randInt" usando 1 como argumento para el parámetro "min" e "i" como el argumento para "max". Para inicializar el generador se requiere de una semilla de 320 bits, que se deriva a partir del valor aleatorio dado por el "pulso de aleatoriedad" obtenido del faro de aleatoriedad de la Universidad de Chile y del nombre de la Junta Electoral que realiza el sorteo. Se detalla más información sobre la semilla en las siguientes secciones.

URI PULSO DE ALEATORIEDAD USADO: <https://beacon.clcert.cl/beacon/2.0/chain/4/pulse/540384>. Valor aleatorio dado por la propiedad "outputValue".

SEMILLA: Semilla usada, en formato hexadecimal:

6db6a00c55de5c34892b43c560e69612c5fc416e2b4eb721e49eb110ff46e6d04e9676c444eeabea

CONSTRUCCIÓN SEMILLA: Se codifica el nombre de la Junta Electoral (en este caso "Macul (Ex Ñuñoa)") según la codificación UTF-8, para obtener su representación como una lista de bytes, la cual se concatena a la derecha de la representación en bytes del valor del pulso aleatorio. La lista de bytes resultante se da como argumento a la función de hash SHA3-512, con lo que se obtiene un hash de 512 bits, de los cuales se usan los primeros 320 como semilla para el generador.

CÓMO USAR NÚMEROS SORTEADOS: Antes de realizar el sorteo, para cada mesa receptora de sufragios, las Juntas Electorales confeccionan una nómina de 30 personas consideradas como posibles candidatas a vocales a partir del padrón de la mesa receptora. Estas nóminas se ordenan alfabéticamente según el nombre de aquellas personas elegidas, escribiendo su nombre completo con los apellidos primero (y siguiendo la forma de ordenar definida en la norma ISO 14651 para el idioma español). Con las nóminas ordenadas, a cada persona se le asigna un número correlativo a su posición en las mismas, de modo que cada número que se obtiene al sortear corresponda al número de alguna persona distinta de la nómina. Entonces, si la mesa receptora necesita nuevos vocales de mesa, las personas sorteadas se designan como tales en el orden en que se sortearon sus números, con las personas sorteadas siendo designadas como vocales reemplazantes.

Si esta es su Junta Electoral, puede visitar la página [¿Cómo le eligieron como vocal de mesa?](#) Allí puede encontrar la nómina usada para su mesa receptora, así como explicaciones generales de todo el proceso.

INFORMACIÓN ELECTORAL

- Leyes**
- Constitución Política de la República d Chile
- Inscripciones Electorales y Servicio Electoral
- Partidos Políticos
- Municipalidades
- Votaciones y Escrutinios
- Gobierno y Administración Regional
- Gasto Electoral
- Tribunales Electorales Regionales
- Tribunal Calificador de Elecciones
- Sistema de Elecciones Primarias
- Disposiciones transitorias leyes 20.901 20.914, 20.915, 20.916
- Partidos Políticos**
- Generalidades**
- Antecedentes Legales**
- I De los partidos políticos, de sus actividades propias y de su ámbito de acción
- II De la constitución de los partidos políticos
- III De la afiliación a los partidos políticos
- IV De la organización interna
- V Del financiamiento
- VI Del acceso a información y transparencia
- VII De la fusión de partidos políticos
- VIII De la disolución de los partidos políticos
- De la disolución de los partidos políticos
- X De los Tribunales y de las normas de procedimiento

Figura 13: Captura de pantalla de toda la página de datos técnicos del sorteo.