Trabajo de Fin de Grado

Grado en Ingeniería Informática

Sistema de votación electrónica basado en blockchain

A blockchain-based electronic voting system

Jesús Eduardo Plasencia Pimentel

La Laguna, 28 de junio de 2018

D. **María Elena Sánchez Nielsen**, con N.I.F. 42.848.599-J profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas, como tutora

C E R T I F I C A

Que la presente memoria titulada:

"Sistema de voto electrónico basado en blockchain"

ha sido realizada bajo su dirección por D. **Jesús Eduardo Plasencia Pimentel**,

con N.I.F. 79.093.085-H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 28 de junio de 2018

Agradecimientos

A mis padres y a mi hermana Andrea. Por el apoyo de todas las formas posibles y por la confianza que siempre me han aportado.

A Víctor, por enseñarme a imaginar y hacer que los años en la facultad pasaran tan rápido.

A Gisela, por acompañarme un rato en descubrirme.

A Valentina, por demasiadas cosas como para escribirlas aquí todas.

Por último, a mi tutora, Elena, por lo ameno que ha sido trabajar con ella y el apoyo que me ha brindado durante la realización de este trabajo.

Licencia

Resumen

Mediante el siguiente trabajo de fin de grado se ha buscado el desarrollo de un sistema de votación distribuido en la red, multiautoridad y de auditoría abierta. Para conseguir dichas características se han hecho uso de tecnologías criptográficas como son el cifrado de umbral, el cifrado homomórfico y la tecnología *blockchain*. Se ha empezado con un análisis del estado actual del tema, mencionando los sistemas ya existentes con sus características importantes, resaltando sus fallos y limitaciones. A partir de un diseño basado en módulos se han elegido las herramientas más apropiadas y se ha implementado un sistema de votación que cumple las características estipuladas siguiendo una metodología iterativa. Procede a la fase de diseño la descripción detallada del proceso de desarrollo en cada una de las partes que lo componen: una sección de protocolos criptográficos; una sección específica a la *blockchain* y sus nodos; y, por último, una dedicada a una aplicación web que sirve de interfaz al sistema. Se realiza una descripción del proceso mediante el cual se puede crear elecciones, crear y entregar papeletas, realizar el escrutinio y verificación de los votos. Tras realizar pruebas experimentales utilizando el sistema desarrollado, se detallan y comentan sus resultados. Se ofrece un comentario de las consideraciones prácticas del voto electrónico y en particular del sistema implementado. Por último se concluye con el planteamiento de líneas futuras para el trabajo realizado.

Palabras clave: evoting, blockchain, criptografía

Abstract

The objective of this project has been to design and implement an efficient voting system that utilizes known cryptography techniques to provide elections that are open-audit, multi-authority and utilize a decentralized network architecture. To fulfill these requirements, threshold cryptography, homomorphic encryption and blockchain technology have been utilized. We have started with a study of the state of the art, commenting already existing systems along with their main characteristics, highlighting their flaws and limitations. Based on a modular design proposal, the appropriate tools have been chosen to implement a voting system that fulfills the chosen characteristics following a iterative design methodology. After the design phase, a section is dedicated to the development of the system structured in three parts: a module that provides cryptographic protocols, a module implementing a blockchain system along with its nodes, and a web application that serves as an interface for the system. Following that, a description of the procedures necessary to hold voting elections is offered, detailing election creation, ballot creation and submission, and tallying and vote verification. After using the system in a real-case scenario, its results are analyzed. We provide commentary on the practical considerations of electronic voting systems and, in particular, the one implemented. Lastly, some conclusions are offered along with recommendations as to the future of this work.

Keywords: blockchain, evoting, cryptography

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

A pesar de la constante digitalización de la vida contemporánea, los sistemas de votación siguen utilizando sistemas físicos que operan con papel. El uso de sistemas electrónicos está comprobado disminuye la abstinencia y favorece el compromiso de la ciudadanía con el proceso político [1]. Esto, sumado con la búsqueda por mayor transparencia en las instituciones públicas, aumenta el interés de incorporar la tecnología *blockchain* al proceso de votación: permite distribuir los datos de la votación entre la ciudadanía para asegurar elecciones transparentes y justas, donde la auditoría puede ser realizada por cualquiera, sin necesidad de recurrir a figuras que deben ser confiadas *a priori*.

Mediante el presente trabajo de fin de grado se ha diseñado e implementado un sistema de votación adaptado a la época contemporánea, a las tecnologías disponibles y las realidades que estas tecnologías permiten crear. Se ha buscado crear un sistema donde no sean necesario autoridades centrales y los datos sean universalmente verificables por cualquiera, donde además la integridad de los datos se mantiene mediante la colaboración de voluntarios que aseguran la integridad del sistema.

Se ha recurrido a la tecnología *blockchain* ya que aporta unas características interesantes: permite crear sistemas donde la confianza es distribuida y no hay un único punto de autoridad. Estas características hacen que esta tecnología sea idónea para diseñar un sistema de votación, como afirma la Unión Europea [2].

Capítulo 2
Diseño

2.1 Caracterización del problema

2.1.1 Objetivos

Se ha planteado el desarrollo de un sistema de voto electrónico que cumpla las siguientes características:

- **Multiautoridad:** no existe una única autoridad del voto. Se pueden organizar elecciones donde las claves privadas no se encuentren en manos de una sola autoridad sino que se encuentren distribuidas de cualquier manera entre múltiples personas.
- **Auditoría abierta:** las papeletas de la votación son públicas y hay mecanismos abiertos a todo el mundo mediante los cuales se puede verificar la validez de las papeletas.
- **Descentralizado en la red:** los datos de la votación –tanto su configuración como los votos emitidos– no existen en un único punto sino que se encuentran distribuidos entre nodos en la red.

Otro objetivo es crear un sistema especializado que no dependiese sobre criptomonedas o *blockchains* ya existentes. Existen ya sistemas de votación usando *blockchain* que utilizan contratos inteligentes (*smart contracts*) para establecer un sistema de votación. Sin embargo, resulta un conflicto de interés respaldar sistemas de uso popular y de importancia política sobre sistemas con una finalidad económica.

2.1.2 Alcance

Para alcanzar los objetivos propuestos se ha estructurado el trabajo de fin

de grado en tres grandes partes:

Un módulo de criptografía, que implementa todos los protocolos necesarios para garantizar la seguridad del sistema. En este módulo se contempla la generación de claves junto con las operaciones de cifrado y descifrado de votos. Además se implementan en este módulo los mecanismos de construcción de pruebas de conocimiento nulo (ZKP, *zero-knowledge proofs*) y de criptografía de umbral (*threshold cryptography*).

Una *blockchain*, junto con sus nodos y los mecanismos que son necesarios para hacerla funcionar. Se contempla aquí los protocolos de validación de la cadena, los mecanismos de replicación, el protocolo de consenso y la creación de bloques.

Por último, una aplicación web que sirve de interfaz con el sistema de voto. Permite la realización del voto y del escrutinio, provee una interfaz de monitorización para la red y permite crear nuevas elecciones. La aplicación web cuenta también con una API que permite interactuar con la *blockchain* mediante mensajes HTTP.

2.2 Estudio y análisis del estado actual

2.2.1 Investigación de sistemas actuales

Existen ya sistemas de votación electrónica que, en su mayoría, se ofrecen como soluciones completas prestadas por empresas privadas. Actualmente, son Smartmatic y Scytl –esta última siendo española– dos de de las más relevantes a nivel internacional. Sin embargo, los sistemas de votación de software libre con uso en casos reales escasean, siendo de especial mención los pocos que reciben el mayor uso: nVotes (anteriormente conocido como Agora Voting), utilizado en las consultas internas del partido político español Podemos; y Helios Voting, desarrollado como un sistema de auditoría abierta, cuyo uso públicamente se limita a elecciones universitarias.

Helios Voting [3] es de particular interés, ya que está basado en el mismo artículo científico en el que se ha basado este trabajo de fin de grado e incorpora múltiples tecnologías que también han sido utilizadas aquí. Una descripción de un caso real del uso de Helios Voting puede ser consultado

en [4]. Posteriormente, se han realizado estudios sobre Helios, corrigiendo pequeñas fallas en su desarrollo [5, 6].

En particular, el campo de sistemas de votación criptográfica sobre *blockchain* es sumamente reciente. A pesar de su poco desarrollo ya se encuentran ejemplos prácticos de su uso en casos particulares [7, 8, 9]; sin embargo, parece ser que no todos son usos honestos [10]. Esto se debe, en parte, a que hay gran interés económico detrás de *blockchain* y cualquier tecnología que pudiera utilizarlo. Las juntas entre sistemas financieros y sistemas de votación no tiene valor ni en el sector público ni en instituciones sin fin de lucro. Sin embargo, este suele ser el tipo que suele imperar en desarrollo e interés, sobre todo a manera de *smart contracts* o sucedáneos implementados sobre criptomonedas [11, 12].

2.2.2 Normativa relevante

Se ha utilizado un documento [13] de Scytl –una empresa especializada en diseño de sistemas de votación– donde detalla la adaptación del sistema propio PNYX al documento [14] de la Unión Europea (UE) que detalla las características que debe tener un sistema de votación electrónico adoptado por cualquier estado miembro de la UE. Los elementos relevantes al sistema desarrollado han sido tomados en consideración, mientras que aquellas partes de la seguridad que se salen fuera del ámbito de este trabajo han sido obviadas.

2.3 Diseño de la plataforma

2.3.1 Contribuciones

En base al estudio de sistemas actuales se plantea coger las mejores partes de cada uno e integrarlas de manera coherente. Se tomará como base el diseño de Helios, junto con las mejoras que se han propuesto posteriormente en [5] y [6]. Aparte, se adaptará el sistema a usar una *blockchain* como sustituto de la *bulletin board* descrita en [16], realizando las decisiones de diseño más apropiadas para adecuar el sistema de votación a esta tecnología.

2.3.2 Elección de herramientas y bibliotecas

A continuación se detallan las elecciones realizadas para llevar a cabo el trabajo de fin de grado, junto con la razón de su uso.

Python 3.6

Python es un lenguaje multiparadigma de uso muy extendido, sobre todo en el campo de la computación científica. Fue elegido para realizar la mayor parte del código por múltiples razones:

1. **Facilidad y velocidad de desarrollo:** es un lenguaje notablemente rápido para el desarrollo iterativo. La gran versatilidad de su librería estándar, sumada con la gran cantidad de librerías de desarrollo de terceros, hace de Python un lenguaje de desarrollo veloz. Se puede también programar sin tener que atender a detalles de implementación a nivel más bajo.
2. **Conveniencia personal:** es el lenguaje de programación que más he utilizado y el que me gustaría usar en mi futuro profesional. Esto me da una mayor motivación para usarlo y potencia aún más su velocidad. Esta comodidad en su uso propicia tiempos de desarrollo más cortos, con código mucho más elegante.

Flask 0.12.2

Flask es un *framework* minimalista de desarrollo de aplicaciones web. Es utilizado para crear una aplicación que permita monitorizar la red e interaccionar con los nodos y la *blockchain* de manera interactiva. Se ha preferido usar Flask sobre Django, una alternativa popular, ya que necesita menos configuración y que para el alcance de la aplicación web no era necesario un *framework* tan completo.

ZeroMQ v4.0 (A través de la librería pyzmq 17.0)

ZeroMQ es una librería de mensajería distribuida que abstrae conceptos de redes de bajo nivel, específicamente TCP y UDP, optando por implementar toda comunicación en un protocolo propio de uso más sencillo. Es utilizada para sustentar la mensajería en la red, permitiendo así efectuar los protocolos de consenso y la comunicación entre nodos. Esta librería maneja detalles importantes de un sistema distribuido seguro, como son la anonimización de los nodos y la seguridad en redes, que permiten entonces

dedicar mayor esfuerzo en la implementación de detalles de más alto nivel.

PyCrypto + PyCryptodome

PyCrypto y su sucesor PyCryptodome son librerías de primitivas criptográficas. Implementar primitivas criptográficas por uno mismo es sumamente peligroso, debido a la facilidad de cometer errores. Con esto en consideración, se ha optado por utilizar una librería que me provee de todas las primitivas que necesito. Así, el peso del trabajo cae sobre la implementación de los protocolos criptográficos de mayor nivel.

Sphinx

Sphinx es una herramienta para la creación de documentación. Permite, a partir de comentarios en el código, generar documentación que después puede ser exportada a múltiples formatos: HTML, TEX, PDF, etc.

2.3.3 Decisión de metodología de trabajo

El enfoque metodológico utilizado fue de **desarrollo incremental e iterativo**. Se definen tareas a realizar que son resueltas mediante un bucle:

1. Se **identifican** los requisitos y el alcance.
2. Se **diseña** una implementación que cumpla los requisitos.
3. Se **implementa** el diseño realizado.
4. Se **realizan pruebas** para verificar el correcto funcionamiento de la implementación.

Esto de manera repetida e incremental. Esta metodología de trabajo permite tener código funcional de manera muy rápida. Así, se puede evidenciar de manera más rápida el progreso hecho ya que siempre se tiene un sistema funcional.

Se ha utilizado también el software de control de versiones Git, ayudado gracias a la página web GitHub en un principio, y posteriormente GitLab. El uso correcto de esta herramienta conlleva a un desarrollo más organizado. Se ha ejercido especial cuidado de organizar todas las tareas a realizar mediante *issues* de la plataforma y de elaborar mensajes de *commit* descriptivos que detallasen el desarrollo del código, que a su vez hacen referencia directa a *issues*, permitiendo tener una bitácora fiel del trabajo

realizado.

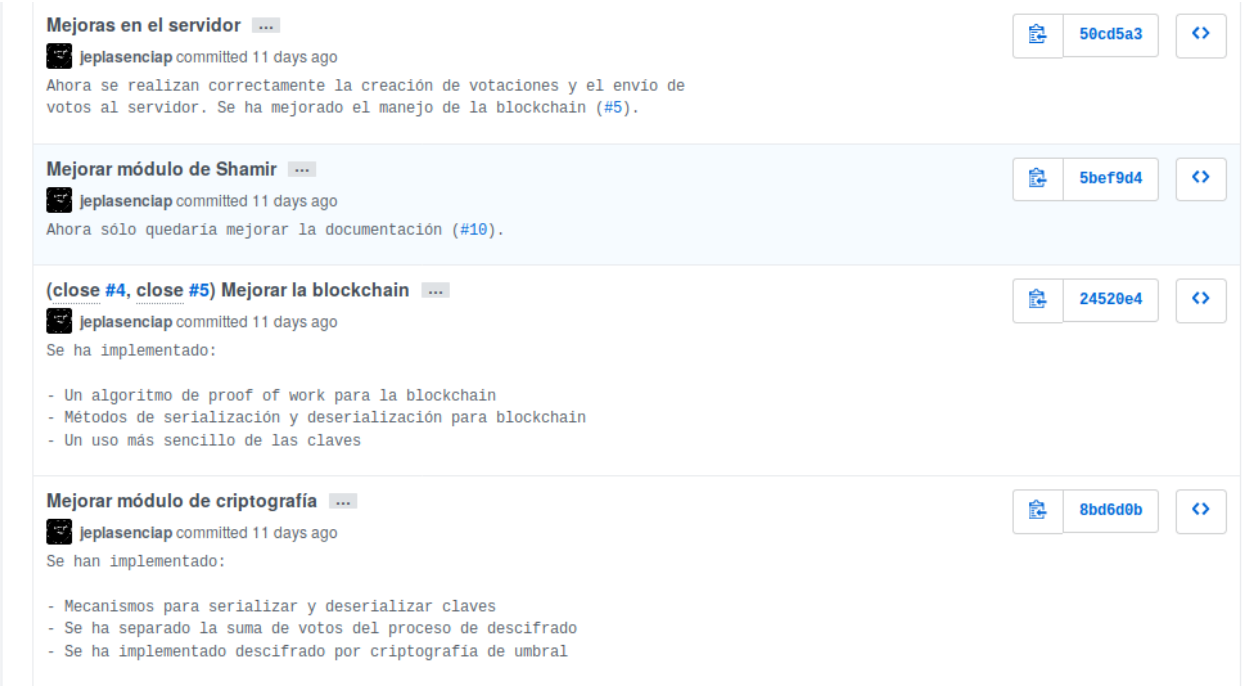

Figura 2.1: *Commits* en la página web Github

Capítulo 3
Desarrollo

Aquí a continuación se desarrolla el procedimiento de desarrollo de la aplicación.

Vínculos de interés:

Repositorio del código	https://gitlab.com/jepp/tfg
Documentación del código	https://jepp.ddns.net

3.1 Planteamiento

Originalmente, el trabajo de fin de grado planteado había sido orientado a un sistema de votación para dispositivos móviles. Desde un principio, la descentralización del sistema de votación había sido un elemento importante en el diseño de la aplicación a realizar. Debido a esto, la tecnología *blockchain* fue escogida: tanto por las características que provee, como también por su relevancia hoy en día. Tras un proceso inicial de investigación y diseño se llegó a la conclusión que compaginar un sistema basado en *blockchain* con un sistema especializado para dispositivos móviles era sumamente difícil, sobre todo si se quería hacer partícipes a dispositivos móviles en respaldar el protocolo de consenso. Hay actualmente ideas descritas en [15] y otros sistemas que funcionan sobre criptomonedas ya establecidas [11, 12], que, como se ha comentado anteriormente, es algo que se busca evitar.

Por lo tanto, se tomó la decisión de enfocar el trabajo de fin de grado a un sistema de votación basado en *blockchain* únicamente. Durante la fase de investigación, se decidió tomar como base la estructura y funcionamiento de

Helios [3], junto con el artículo en el que se basa su diseño [16], tomando medidas especiales para adaptarlo a la idea de una *blockchain*.

3.1.1 Decisiones técnicas

Para acotar debidamente el alcance del trabajo de fin de grado, se han tomado dos grandes decisiones frente a su desarrollo.

Para facilitar el despliegue de la aplicación se utilizará un servidor central desde el cual corre una aplicación web que sirve de intermediario entre nodos. De esta manera se elimina la necesidad de implementar mecanismos de descubrimiento de nodos y así se agiliza el desarrollo. Esto implica que el desarrollo de la red P2P será limitada, puesto que un mayor desarrollo aumentaría el alcance considerablemente.

Otro elemento que no está contemplado bajo este trabajo de fin de grado son los procesos de generación conjunta y de distribución de pares de clave pública-privadas o de firmas digitales, el uso de *mixnets* para anonimización y otros temas de seguridad que son periféricos al sistema en sí. Sin embargo, sí se encuentran implementados todos los métodos necesarios para generar claves, cifrar, descifrar, firmar, etc.

Aparte de estas dos decisiones, es necesario comentar que en el desarrollo del código se ha optado por obviar ciertos detalles que no son explícitamente necesarios para el funcionamiento básico del código, pero son recomendables o hasta necesarios de implementar de utilizarse el sistema desarrollado en un entorno real. Cabe mencionar, entre otras cosas, la necesidad de implementar mecanismos de validación más robustos; implementar partes del código de lenta ejecución en un lenguaje de nivel más bajo para conseguir mayor velocidad, como puede ser el uso de C combinado con las librerías ctypes o Cython para integrarlo con Python; y mejorar la presentación gráfica de la aplicación web, incluyendo adaptarla para dispositivos móviles.

Por último, el código desarrollado está publicado bajo la *Affero GNU Public License v3.0* (AGPLv3.0), una licencia de software libre, copyleft, diseñada especificamente para asegurar cooperación con la comunidad en caso de software que se ejecuta desde un servidor.

3.1.2 Serialización

Un elemento importante a mencionar en el desarrollo de los diferentes módulos del código es cómo codificar la información para transmitirla a través de redes y para guardarla en medios de almacenamiento. Se ha optado por serializar toda la información en formato JSON, debido a su facilidad de uso en Python y la facilidad con la que se puede leer e interpretar por humanos. Para una implementación en un entorno real es importante declarar una serialización más compleja –a nivel de bits– para ahorrar en espacio y tiempo, aparte de proveer mecanismos de integridad como puede ser *checksums*.

3.2 Documentación

La documentación ha sido realizada acorde a las reglas de documentación delineadas en dos documentos oficiales de Python: PEP8[17], el estándar para estilo de código, y PEP257[18], el estándar para documentación de código. La generación de la documentación fue realizada mediante la herramienta Sphinx, que genera páginas HTML a partir de las cadenas de documentación presente en el código fuente. La documentación es automáticamente generada después de cada cambio en el repositorio de gitlab gracias a la funcionalidad de *continuous integration* (CI, integración continua) que provee dicha página. Tras cada *push* al repositorio se ejecuta un *script* que vuelve a generar la documentación y actualiza la página para reflejar los cambios. Este script puede ser consultado en el fichero `.gitlab-ci.yml`.

3.3 Criptografía

El módulo que más trabajo ha llevado –sobre todo de investigación–, es el de criptografía. El protocolo utilizado se basa en el descrito en [16]. Aquí adelante se detalla el sistema implementado, las razones para su diseño y cómo ha sido utilizado.

3.3.1 Criptosistema

Se ha utilizado el esquema de cifrado ElGamal, en específico la variante basada en grupos cíclicos. Es también posible implementar el sistema propuesto con la variante que utiliza curvas elípticas. A continuación se

ofrece una explicación sencilla sobre ElGamal basado en grupos cíclicos.

ElGamal es un criptosistema de clave asimétrica basado en el intercambio de claves Diffie-Hellman. Su seguridad está basada en la dificultad del cálculo del logaritmo discreto en un grupo cíclico. Los pares de claves pública-privada se caracterizan por unos parámetros (p, g, x, y) de un grupo cíclico G. El parámetro p es el orden del grupo, que debe ser un número primo que cumpla unas ciertas características; g es un generador del grupo; x es la clave privada e y es la clave pública.

ElGamal es el criptosistema utilizado en los sistemas de votación de software libre estudiados ya que es el criptosistema sin patente más eficiente que todavía cumple las características necesarias para un sistema de votación: es de cifrado probabilista, es parcialmente homomórfico y se presta a técnicas de cifrado de umbral. Una alternativa posible es el sistema Paillier que no se suele utilizar porque todavía se encuentra bajo patente y no es tan rápido.

3.3.2 Cifrado homomórfico

Se dice que un sistema de cifrado es homomórfico cuando permite realizar operaciones matemáticas sobre texto cifrado tal que, al descifrarlo, el texto descifrado resultante se vea afectado de manera esperada por las operaciones realizadas. Para el sistema desarrollado es necesario sumar los votos todavía cifrados. Para esto necesitamos un esquema de cifrado que sea homomórfico en adición, que permita que datos cifrados se puedan sumar entre sí sin necesidad de descifrarlo.

El sistema criptográfico escogido, ElGamal, es homomórfico en multiplicación: $D(C(x\times y))=D(C(x)\times C(y))$, donde C es la función de cifrado y D la de descifrado. Pero, atendiendo a la propiedad de la potenciación que afirma $x^a\cdot x^b=x^{a+b}$, podemos utilizar este esquema como un sistema homomórfico en adición. Para esto se procede a hacer lo siguiente: para sumar dos números a y b, se cifra g^a y g^b, siendo g un generador; en nuestro caso, nos vale el generador perteneciente al dominio de la clave. Al realizar el producto entre el texto cifrado de g^a y el de g^b conseguimos el texto cifrado de g^{a+b}. Al descifrarlo hay que conseguir $a+b$ a partir del cálculo del logaritmo discreto de $a+b$ base g, un problema considerado difícil, debido a que su tiempo de ejecución es exponencial en el número de dígitos del grupo cíclico G al que pertenece el generador g del dominio de

la clave.

Al ser los posibles valores de $a+b$ números relativamente pequeños, acotados por el número de votantes, existen opciones eficientes para calcular el logaritmo discreto. Se puede usar un algoritmo para la obtención de logaritmos discretos, como también se puede computar una *lookup table*. El tiempo que toma generar esta *lookup table* es polinómico en el número de votantes, añadido a que puede empezar a ser generada desde que se genera la clave. Por lo tanto, se ha optado por esta última opción en vez de implementar un algoritmo de obtención de logaritmos discretos. Sin embargo, es relevante mencionar que los dos algoritmos exactos más útiles para esta tarea son el algoritmo *baby-step giant-step* y el algoritmo de Pohlig–Hellman, que consiguen el mismo objetivo, pero requieren saber de antemano el valor del texto cifrado de la suma final de los votos. El algoritmo *baby-step giant-step* se ha implementado a forma de demostración y para permitir también hacer comparaciones de tiempo.

La *lookup table* es computada de la siguiente manera: se produce una lista donde cada elemento es igual a $g^i \bmod p$, con i desde 1 hasta el número de votos válidos totales (el número máximo que puede haber para una misma opción). Para conseguir el valor de un número cifrado g^x se busca su valor en la lista y el índice del valor dentro de la lista da el valor de x.

3.3.3 Cifrado de umbral

El cifrado de umbral ha sido implementado mediante el algoritmo *Shamir's Secret Sharing*. La clave privada de la votación es dividida en *shares*, partes únicas que juntas pueden reconstruir la clave. Se ha implementado de tal forma que la clave se divide en un número n de *shares*, de los cuales un número k de *shares* son necesarios para reconstruirla. Cabe mencionar que es necesario algún mecanismo de computación distribuida para descifrar con seguridad la suma final de votos, ya que reconstruir la clave privada no es seguro. Sin embargo, esto no está contemplado en el código realizado. A continuación se da una explicación sencilla del algoritmo.

El algoritmo se fundamenta en una propiedad de los polinomios: son necesarios **como mínimo** $n+1$ puntos para definir un polinomio de grado n. Cuando se conocen solamente n o menos puntos, hay infinitos polinomios de grado n que pasan por dicho conjunto de puntos. Atendiendo

a esta propiedad, es posible dividir una clave privada del esquema ElGamal en n *shares*, donde, como mínimo, k *shares* son necesarios para reconstruir la clave de la siguiente manera:

Se genera un polinomio de grado k donde el término independiente del polinomio es igual a la clave privada. Hay que primero escoger un número primo m mayor que todos los términos del polinomio, para esto se ha escogido un número primo de Mersenne (un primo de la forma 2^x-1) mayor que la p de la clave. Para una clave de 2048 bits, escogemos $2^{2203}-1$. El resto de términos del polinomio son números $\{x_1, x_2, \ldots, x_{k-1}\}$, $0<x_i<m$ generados al azar. Se calculan n puntos del polinomio: $(1,f(1)),(2,f(2)),\ldots,(n-1,f(n-1)),(n,f(n))$. El punto $(0,f(0))$ es la clave privada. Cada *share* entonces es una tupla $(i,f(i)),1\leq i\leq n$. De esta manera, con **por lo menos** un número k de *shares* será posible reconstruir la clave privada, ya que es posible identificar de manera única el polinomio.

Para reconstruir la clave privada a partir de las *shares* se utiliza la interpolación lagrange de polinomios. Se utiliza una fórmula para reconstruir el polinomio basado en puntos conocidos del polinomio, en este caso las *shares*. Habiendo reconstruido el polinomio, el valor de la clave privada es el término independiente o, lo que es lo mismo, *f(0)*: el valor de evaluar el polinomio para $x=0$.

3.3.4 Pruebas de conocimiento nulo

Las pruebas de conocimiento nulo o pruebas de conocimiento cero (ZKP, *zero-knowledge proofs*) son pruebas criptográficas que permiten a una de las partes de un protocolo criptográfico demostrar que conoce un valor dado x sin revelar ni el valor de x ni ninguna otra información adicional.

Utilizando pruebas de conocimiento nulo aseguramos que las papeletas contengan votos válidos sin revelar absolutamente nada del valor de las papeletas. Se da por válido un voto que cumpla las siguientes características:

1. El voto es una lista de valores ordenados que pueden tomar únicamente el valor 0 o 1.
2. La suma de los valores de la lista que representa el voto puede ser 0 o 1, siendo 0 un voto nulo.

Además, para fines de verificación posterior, necesitamos que la prueba de

conocimiento nulo sea no interactiva, dígase: no requiere interacción alguna entre las partes del protocolo, la que demuestra y la que verifica. Para esto se utiliza la heurística Fiat-Shamir, que consiste en utilizar una función *hash* para servir de generador de desafíos para la prueba.

Se ha implementado la prueba criptográfica llamada *Disyunctive Chaum-Pederson Protocol* (DCP). La prueba DCP permite afirmar que un valor cifrado es definitivamente uno de dos valores x o y. En nuestro caso los dos valores posibles son 0 o 1, o más correctamente, atendiendo a lo descrito en la sección de cifrado homomórfico: g^0 o g^1. Una definición completa del protocolo se puede encontrar descrita en [6].

3.3.5 Firmas digitales

Para las firmas digitales se ha utilizado ECDSA, un algoritmo de firma electrónica utilizado en Bitcoin y en la mayoría de criptomonedas.

Para firmar una papeleta se firma con la clave privada el texto resultante de aplicar la función *hash* SHA256 a la clave pública correspondiente. Para verificar que la firma para una papeleta es válida se verifica con cada clave pública en la lista de votantes hasta dar con la correcta. Si ninguna clave pública en la lista de votantes puede verificar la validez de la firma se entiende que el voto no es válido. Se recomienda generar un par de claves pública-privada diferente para cada elección a fines de seguridad.

Sólo contará como válida la última vez que se utilice una firma en la votación. Esto con el fin de permitir la modificación del voto.

3.3.6 Implementación de la criptografía

Para implementar el módulo criptográfico fueron utilizadas las librerías PyCrypto y PyCryptodome, que implementan primitivas criptográficas de bajo nivel y manejan detalles de bajo nivel tales como evitar *side-channel attacks* y la optimización de los mecanismos de creación de claves y cifrado-descifrado.

3.4 Blockchain

Se ha desarrollado una *blockchain* sencilla que implementa todas las funciones básicas: creación de transacciones (votos, en este caso), creación de bloques, verificación de validez y actualización.

3.4.1 Descripción básica de una *blockchain*

Una *blockchain* es una herramienta que permite mantener múltiples copias de un conjunto de datos sincronizadas entre sí. Un nombre que también se le da es «distributed ledger», esto es, libro de consultas distribuido. La *blockchain* se estructura en bloques ordenados (de ahí su nombre: «cadena de bloques»), donde a su vez, cada bloque está compuesto por unidades de almacenamiento (en esto caso, votos), que contienen los datos que se buscan guardar y distribuir.

En la mayoría de las *blockchain* el mecanismo que asegura la integridad de los datos es conocido como *proof-of-work* («demostración de trabajo»): busca que un cierto tiempo de procesador haya sido invertido en garantizar que los datos sean correctos. Los participantes en la validación de la *blockchain* reciben notificaciones de nuevos votos realizados mientras constantemente buscan un *proof-of-work*. Cuando se consigue uno válido los votos realizados hasta ese entonces son guardados en un nuevo bloque y la búsqueda por el nuevo *proof-of-work* empieza. Si hay una cantidad mayoritaria de nodos participantes actuando honestamente el sistema se mantiene seguro.

3.4.2 Estructura de la *blockchain*

La *blockchain* implementada posee una estructura muy sencilla, de tal manera de no complicarse en detalles de implementación. Una *blockchain* más robusta, apropiada para uso en casos reales conllevaría una estructura más complicada, especificando, por ejemplo, cómo es la representación de la cadena a nivel de bits.

El primer bloque de la *blockchain* es conocido como el bloque génesis (*genesis block*) y posee unas características únicas frente a los otros bloques. Principalmente, no almacena transacciones, sino que su función es almacenar información sobre la cadena en sí. La clase cadena implementada posee un bloque génesis con la siguiente información:

- **Índice:** El índice del bloque génesis es 0.
- **Prueba (*proof*):** Un primer *proof-of-work* generado al azar, para permitir la verificación de bloques posteriores.
- **Timestamp:** El tiempo de creación de la cadena, necesario para asegurar la validez de la cadena.

- **Tiempo de inicio:** El tiempo a partir del cual nodos honestos aceptan votos.
- **Tiempo de finalización:** El tiempo a partir del cual nodos honestos dejan de aceptar votos.
- **Clave pública:** La clave pública utilizada para cifrar los votos y construir las pruebas.
- **Lista de votantes:** la lista de claves públicas que pueden verificar la validez de las firmas digitales emitidas.
- **Lista de opciones:** una lista ordenada de las posibles opciones como cadenas sencillas. A manera de ejemplo la lista [“María”, “Juan”, “Sofía”], donde [0,0,1] sería un voto por Sofía.
- **Nombre:** El nombre de la elección.

El resto de bloques almacenan información. Su estructura es la siguiente:

- **Índice:** la posición del bloque dentro de la cadena.
- **Timestamp:** su tiempo de creación.
- **Prueba (*proof*):** el proof-of-work obtenido para este bloque.
- **Hash anterior:** un hash del bloque anterior en la lista.
- **Transacciones:** las transacciones, en este caso papeletas de voto, que han sido almacenadas en el bloque.

Por último cada papeleta se estructura de la siguiente manera:

- **Options:** el texto cifrado del voto.
- **Proofs:** las pruebas criptográficas del voto.
- **Signature:** la firma digital.

3.4.3 Proof-of-work

El protocolo de consenso escogido es uno sencillo similar a *hashcash*, el algoritmo de *proof-of-work* adaptado para Bitcoin. Se ha elegido

implementar un algoritmo sencillo para ahorrar en tiempo. A continuación se presenta el algoritmo implementado.

Consiste en obtener un número que cumpla la siguiente característica: aplicando una función *hash* sobre la concatenación del *hash* del último bloque, la última prueba exitosa y el número, el *hash* resultante empiece por una cantidad determinada de ceros. Los diferentes nodos compiten por conseguir dicho número, empezando por un número aleatorio, que va aumentando hasta conseguir uno que cumpla el requisito.

La dificultad de conseguir una prueba útil está determinada por la cantidad de ceros que se pida del texto resultante de la función *hash*. La función *hash* escogida es SHA256, debido a que se encuentra ya implementada en la librería estándar de Python, es rápida y se considera segura al día de hoy.

Este sistema no es el ideal para un sistema de votación, sin embargo, fue escogido para simplificar la implementación del sistema. En la sección de consideraciones prácticas se plantean sistemas alternativos que se consideran mejores que el implementado.

3.4.4 Verificación y validación

La *blockchain* se mantiene válida mientras el trabajo de procesador invertido por los nodos honestos supere con un cierto margen el de los nodos no honestos. De esta manera los bloques contienen la información únicamente de votos aceptados como válidos por nodos honestos. Por esto, la *blockchain* más larga donde todas las pruebas sean correctas es siempre la válida.

También es necesario que todos los bloques estén ordenados linealmente en el tiempo, esto es: todo bloque tiene una firma sobre su timestamp que sólo puede ser válida si ese bloque fue creado después del bloque anterior y antes del bloque siguiente. Esto hace que reemplazar un bloque en la mitad de la cadena sea imposible o que, por lo menos, implicaría cambiaría ese bloque y todos los demás de él también, teniendo que generar pruebas válidas para cada uno. Realizar esto requiere un esfuerzo computacional enorme y por tanto es inviable al menos que un atacante posea un poder computacional muy elevado.

3.4.5 Nodos

Los nodos han sido implementados mediante ZMQ, una librería de mensajería distribuida. Los nodos componen la red P2P del sistema y son quienes se encargan de verificar la integridad de la información sobre la *blockchain* realizando el *proof-of-work*.

Para procesar los mensajes los nodos trabajan con dos puertos: uno para nodos y otro específico para el servidor. Los nodos envían mensajes con entrega de mejor esfuerzo (*best-effort delivery*), por lo tanto no hay ninguna garantía de que el mensaje sea entregado. Los mensajes entre nodos incluyen mensajes que permiten unirse a la red, mensajes que declaran votos, mensajes que inician elecciones y mensajes que piden copias de la *blockchain* a los nodos que conocen. Todos los nodos tienen su propia lista de nodos conocidos y *blockchains* conocidas.

Ya que la comunicación es *blocking*, esto es, detiene el hilo que la ejecuta, la implementación de los nodos se hizo mediante un sistema multihilos: hay un hilo encargado de recibir mensajes y procesarlos, y otro hilo encargado de manejar una cola de mensajes por mandar, envíandolos a los nodos que conoce.

3.5 Aplicación web

La aplicación corre sobre un servidor web que funciona con Flask. El *front-end* permite interactuar con el sistema de voto, junto con monitorizar los eventos del servidor y de la *blockchain*. El *back-end* maneja todos los detalles necesarios para crear elecciones y votos, recibir y manejar votos, y monitorizar la red en tiempo real.

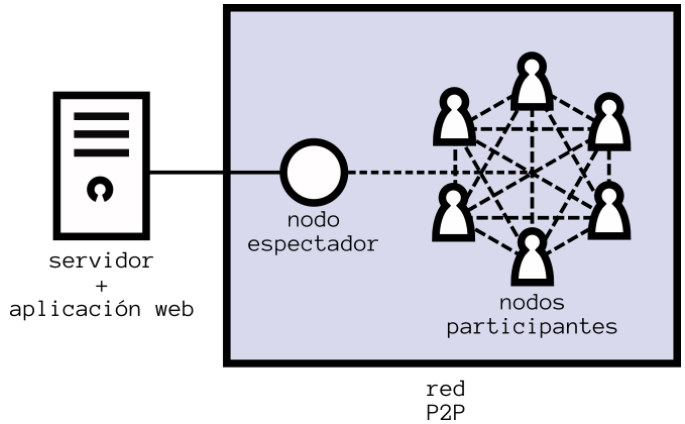

Figura 3.1: Estructura de la red P2P y la aplicación web en conjunto.

3.5.1 Interfaz con el voto

Se han diseñado tres páginas sencillas que permiten las operaciones básicas: la creación de votaciones, la emisión de votos y el escrutinio de votaciones finalizadas.

Las páginas han sido diseñadas mediante la herramienta Jinja2 que acompaña a Flask. Permite crear plantillas HTML que después son rellenadas con datos generados en Python, creando páginas web dinámicas.

3.5.2 Monitorización de la blockchain

Mediante este módulo se puede monitorizar en tiempo real los eventos del servidor y de la blockchain: la emisión de votos, la creación de bloques, el inicio y final de votaciones y también información adicional del servidor.

La implementación de este módulo se ha conseguido mediante socket.io y Flask-socketio, que permiten establecer una comunicación bidireccional en tiempo real entre el servidor y el cliente de la aplicación web. Para esto, se registran *hooks* en Flask: funciones *callback* que se ejecutan cuando se reciben mensajes del cliente.

La monitorización está sustentada por un nodo espectador de la red, que recibe los mensajes transmitidos a través de ella y una conexión a través de XHR-polling entre el navegador del usuario y el servidor. Los mensajes recibidos son convertidos en mensajes legibles para el usuario y actualizados en la página.

3.5.3 API

Se ha implementado una API mediante consultas POST al servidor que permite interactuar con la *blockchain* a través de una serie de rutas:

`/api/templates`: devuelve una lista de plantillas a rellenar para el resto de consultas API donde se detalla la estructura del objeto JSON que espera la ruta y el formato del mensaje que devuelve, tanto en casos de éxito como en casos de error.

`/api/create`: para crear votaciones.

`/api/cast`: para emitir una papeleta de voto.

`/api/tally`: para realizar un escrutinio de votos.

`/api/log`: para recibir un registro de los últimos eventos del servidor y la blockchain.

La implementación del API se ha llevado a cabo mediante la funcionalidad de rutas de Flask, donde las rutas admiten sólo objetos JSON correctamente formados, que después ejecutan funciones en el servidor. El contenido de todos los mensajes es parseado y saneado para evitar fallos de seguridad.

3.5.4 Seguridad

Al ser una aplicación web se ha tenido en cuenta también detalles de seguridad tales como posibles vulnerabilidades CSRF y XSS. Se ha optado por mantener la aplicación web lo más sencilla posible para evitar posibles vulnerabilidades y se ha recurrido a la propia funcionalidad de Flask para sanear todos los elementos que van a ser insertados en la página web de manera dinámica. La información guardada y cargada por la página web es siempre almacenada en un formato seguro para la web, con todos los escapes de caracteres HTML necesarios.

No se han tomado pasos activos para combatir CSRF puesto que la aplicación web no tiene la complejidad necesaria para ameritarlo, pero de ser implementado el sistema en una página más compleja, como por ejemplo una que tenga un sistema de login, es recomendable el uso de WTForms o un *framework* más robusto de desarrollo web, como Django, que ya tienen mecanismos propios de defensa contra CSRF.

3.5.5 Capturas de la aplicación web

Ya que la aplicación web sirve para realizar pruebas no se empleó particular cuidado en su presentación más allá del básico. Se ha utilizado Bootstrap para realizar páginas web con diseño responsivo de tal manera que puedan ser utilizadas en dispositivos móviles y ordenadores por igual. A continuación siguen capturas de diferentes páginas de la aplicación web:

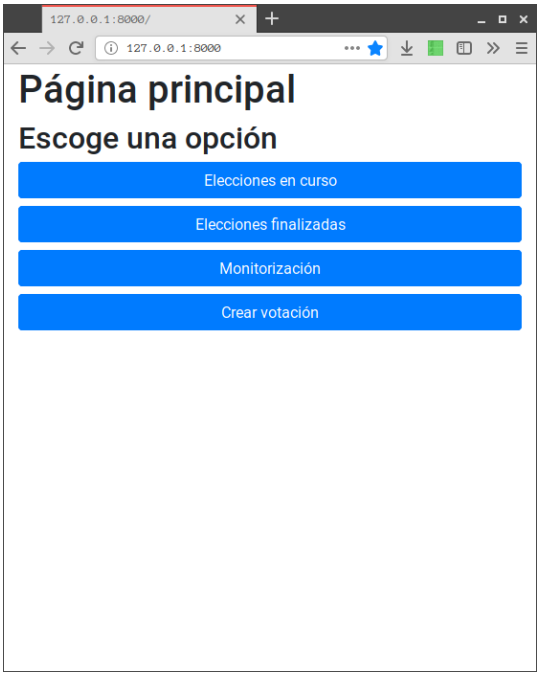

Figura 3.2: Página principal de la aplicación web.

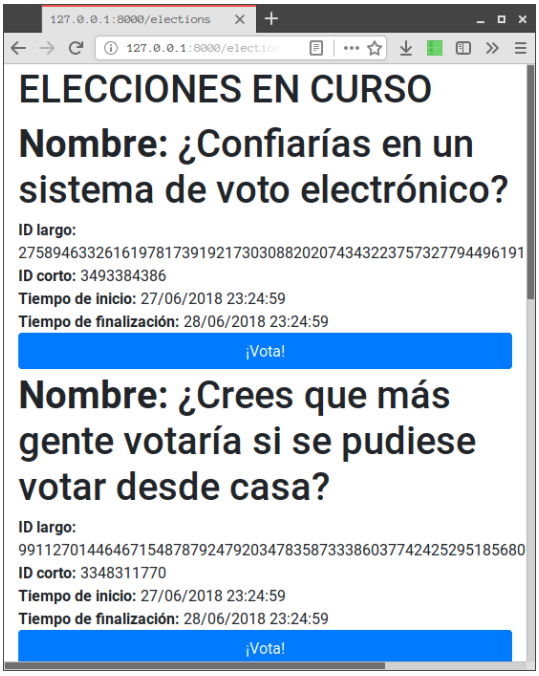

Figura 3.3: Página de elecciones en curso.

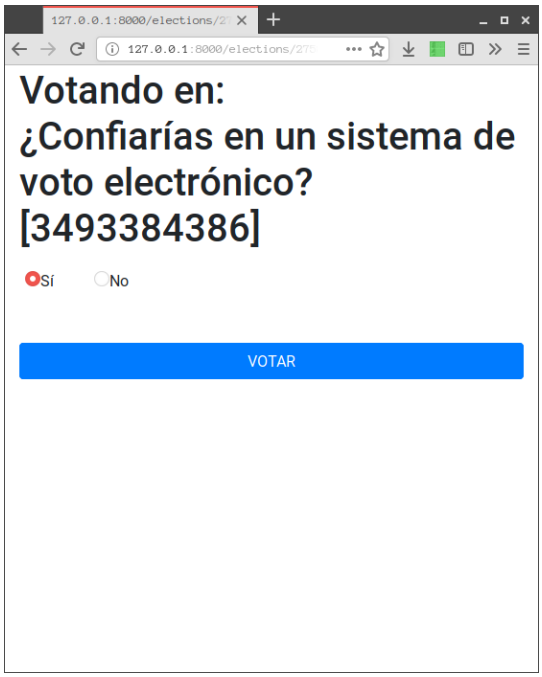

Figura 3.4: Página de elección de opciones para una elección.

Figura 3.5: Confirmación antes de votar definitivamente.

Figura 3.6: Un comprobante de voto.

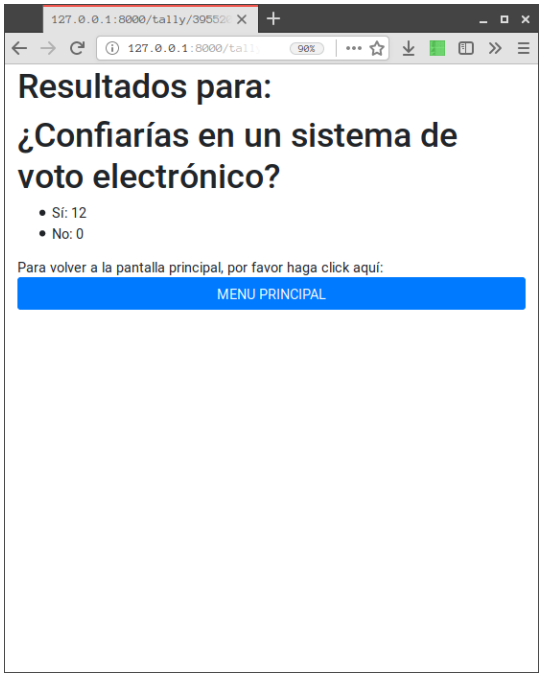

Figura 3.7: Los resultados de una elección.

3.6 Problemas encontrados

Aquí a continuación se detallan los mayores problemas encontrados, junto con cómo se ha enfrentado su solución.

La aplicación web y los nodos ambos tienen implementaciones multihilos. En particular corre sobre los llamados hilos verdes (*green threads*): hilos implementados a nivel de *user space* del sistema operativo, esto es, no dependen de la implementación de hilos del *hardware*. La librería utilizada para el manejo de hilos ha sido Eventlet. El uso de hilos verdes ha sido necesario, puesto que hay ciertas partes del sistema, a mencionar los nodos y el servidor, que los necesitan para funcionar de manera correcta. Se ha procurado realizar una implementación sencilla del sistema multihilos, sin embargo, la programación multihilos conlleva cierta dificultad y ha llevado a dos problemas:

- La aplicación web no es capaz de actualizar en vivo. Normalmente, Flask es capaz de cambiar código en caliente sin necesidad de reiniciar el servidor que está sirviendo la aplicación. Sin embargo, la implementación de hilos realizada producía problemas con esta

funcionalidad y el servidor implementado necesita reiniciar para poder cambiar el código que ejecuta.

- Los nodos no tienen forma de cerrar limpiamente, ya que los hilos que corre no tienen implementado un mecanismo que permita detenerlos, así que la única forma de cerrar un nodo es matar el proceso que lo mantiene.

Capítulo 4
Procedimiento del voto

Mediante los siguientes apartados se detallará los procedimientos necesarios para crear una elección, que implica también pautar su fecha y hora de comienzo y de finalización, establecer las posibles opciones y el mecanismo de firma; como también cómo se vota, cómo se escrutan los votos de una elección finalizada y cuáles son los mecanismos que aseguran la validez de cada voto y de la elección en su totalidad.

4.1 Creación de elecciones

El primer paso para votar es crear una elección. Esto puede ser realizado a través del servidor o a través de algún nodo que participe en la red. Se genera una nueva *blockchain* cuyo bloque génesis contiene la configuración de la elección. Dicho bloque es transmitido por la red de tal manera que todos los nodos lo reconozcan. A partir de este momento ya es posible ir generando la *lookup table* que permite saber los resultados finales de la votación.

Las votaciones están caracterizadas por:

- **ID:** un UUID que identifica únicamente a la elección.
- **Nombre:** una cadena que nombra a la votación, haciéndola reconocible de manera humana. Por ejemplo: “Elección para presidenta de asamblea”.
- **Tiempo de inicio:** el tiempo a partir del cual se aceptan votos en la elección. Es diferente del tiempo de creación de la elección, que indica cuando fue creada. Es utilizado para programar el inicio de una elección.

- **Tiempo de fin:** el tiempo de cierre de la elección. No se aceptarán votos después de este tiempo y a partir de que se alcance este momento es posible escrutar los votos.
- **Lista de opciones:** las posibles opciones a elegir.
- **Lista de votantes:** los votantes autorizadas a participar en la elección

4.2 Votación

Las papeletas de los votos son creadas desde los nodos o desde el servidor y distribuidos a todos los participantes de la red. Los nodos con buen comportamiento rechazarán de inmediato papeletas mal formadas. La papeleta de voto equivale a una transacción en la *blockchain* y desde que es reconocida por un nodo, éste la incluirá en el próximo bloque a ser generado.

Los votos se estructuran de la siguiente manera:

- **Voto:** es una lista del tamaño de las opciones donde cada elemento es un texto cifrado de g^0 o g^1, donde como máximo hay un g^1, siendo el resto g^0. El índice del valor cifrado de g^1 dentro de la lista indica por qué opción se ha votado.
- **Pruebas criptográficas:** una lista del tamaño del número de opciones más uno de pruebas DCP para el voto. Verifican que sólo hay 0s y 1s en el texto plano de la papeleta y que cómo máximo hay un sólo 1 entre ellos. El número de pruebas es igual al número de opciones más uno.
- **Firma:** la firma que asegura que el emisor del voto está autorizado a votar en la elección.

4.3 Verificación y escrutinio

Al finalizar la votación, ningún nodo honesto aceptará como válido un voto emitido. Se esperará a que se cierre el último bloque con los últimos votos emitidos antes de poder escrutar los votos.

El escrutinio se realiza en múltiples partes:

1. Se recorre la *blockchain*, quedándose con sólo aquellos votos que sean válidos, esto es: que tengan una papeleta del tamaño correcto, que sus pruebas sean válidas y que su firma digital esté autorizada por la lista de votantes.

2. Se realiza el productorio de los textos cifrados de los votos válidos. El resultado entonces será una lista de elementos del tamaño de las opciones cuyo texto cifrado corresponde a g^x, donde x es la suma de votos para esa opción.

3. Se descifran los elementos de la suma final, dándonos el valor g^x para cada opción. En este momento hay que calcular el logaritmo discreto de cada elemento para conseguir la suma de votos de cada opción. El logaritmo discreto se obtiene mediante una consulta en la *lookup table* correspondiente a la clave.

Para escrutar los votos es necesario contar con la clave privada o con los *shares* suficientes de la clave privada como para efectuar el descifrado de la suma final, que se realiza mediante una computación compartida: un algoritmo que asegura que sólo se va a usar la clave para descifrar el voto final y hace que no sea necesario reconstruir la clave. Existe también un algoritmo que asegura que se ha descifrado correctamente los votos mediante pruebas de conocimiento nulo. Este último algoritmo junto con el algoritmo para realizar la computación distribuida no fueron implementados en el código de este trabajo de fin de grado, pero se encuentran descritos en [16] y [6].

Capítulo 5
Resultados experimentales

Para comprobar el funcionamiento del sistema en un entorno real se organizó una encuesta a través de votaciones en la aplicación web. Las preguntas fueron las siguientes:

1. ¿Crees que un sistema de votación descentralizado, sin autoridad única y de auditoría abierta podría ser útil para instituciones públicas y asociaciones tales como sindicatos y cooperativas?
2. ¿Crees que más gente votaría si se pudiese votar desde casa?
3. ¿Confiarías en un sistema de voto electrónico?
4. ¿Te suena el término «*blockchain*»?

Las preguntas fueron escogidas para establecer una idea del conocimiento de la persona normal en sistemas de votación y su grado de confiabilidad en sistemas de votación electrónico.

Se desplegó la aplicación web sobre un servidor casero corriendo sobre una Raspberry Pi. Se configuró un dominio mediante un servicio gratuito de DNS dinámico para dirigir tráfico al servidor desde una URL fácil de recordar: jepp.ddns.net:8000. Sobre este mismo servidor corrían nodos que corrían algoritmos de *proof-of-work* para crear bloques en la *blockchain*.

Figura 5.1: Los mensajes de actualización de bloques en el servidor

Para los fines de esta votación y para que pudiese votar la mayor cantidad de personas se optó por no distribuir firmas digitales sino por utilizar la combinación de la dirección IP con el *user agent* del navegador del usuario para hacer el voto único por dispositivo y por dirección IP, siendo este el mecanismo utilizado en la mayoría de páginas web de encuestas sencillas anónimas.

Las preguntas se pautaron iniciar a las 12:00 y acabar a las 22:00 de un mismo día. Los usuarios han sido personas de edad universitaria que han accedido tanto desde móvil como desde ordenadores sobremesa o portátiles. Durante el tiempo de realización ningún usuario reportó irregularidades y el código corriendo en el servidor fue actualizado múltiples veces sin problemas. No se perdieron votos y los votos repetidos fueron correctamente contados como tal.

Finalizado el tiempo pautado para la encuesta se escrutaron los votos utilizando directamente la clave privada de la votación. Los resultados se encuentra comentados a continuación:

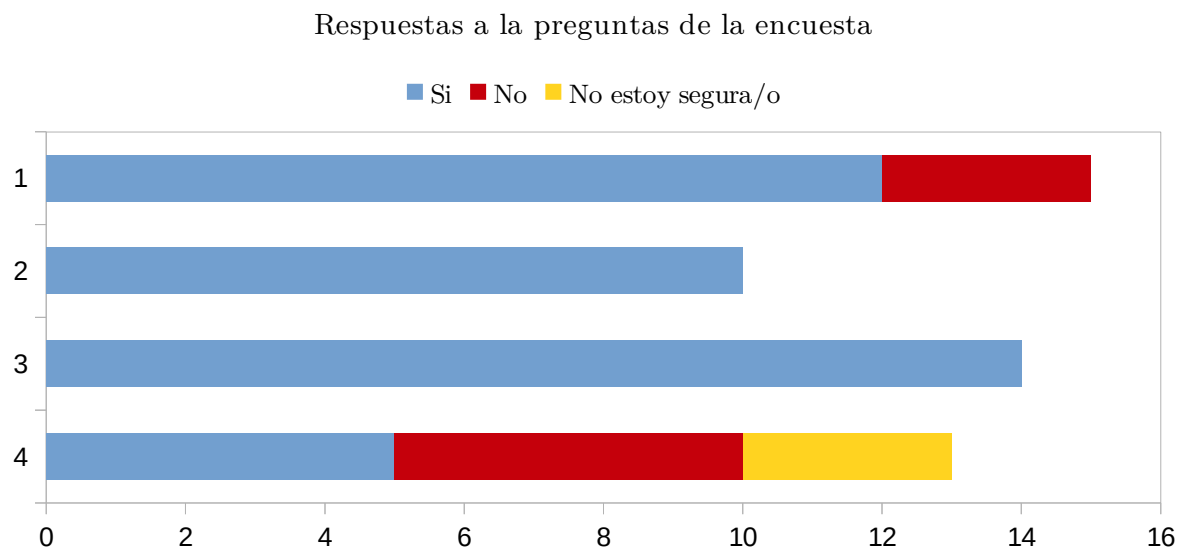

Figura 5.2: Resultados de la encuesta

La disparidad en la cantidad de votos se debe a que algunas personas sólo contestaron una pregunta ya que para elegir otra pregunta debían volver desde el menú principal. Un mejor diseño para la página web –más especializado para el caso particular– podría haber remediado este problema.

Se considera que su uso experimental ha sido exitoso y que este sistema puede ser adaptado –con unos pocos cambios– para ser usado dentro de múltiples contextos.

Capítulo 6
Consideraciones prácticas

6.1 Escalabilidad

Debido a que el número de votantes puede variar desde un grupo de amigas decidiendo la pizza que van a comprar para cenar hasta una elección nacional vinculante, un aspecto importante a analizar del sistema es su escalabilidad: cómo cambia el desempeño del sistema en función del número de participantes en él.

El tamaño del voto varía en función del tamaño de la clave escogida y del número de opciones posibles que hay en el voto (esto es debido a que son necesarias $n+1$ pruebas criptográficas para cada papeleta de n opciones), siendo la siguiente la fórmula para calcular el tamaño de cada voto en bits, asumiendo una forma óptima de almacenamiento:

$$2tn+(n+1)(8t)+s$$

Donde n es el número de opciones posibles, t es el tamaño de la clave en bits y s es el tamaño de la firma en bits. Podemos ver que el tamaño de la papeleta es lineal en el número de opciones y en el tamaño de la clave. Con estos datos es posible calcular algunas cifras orientativas de cuánto pueden pesar en disco los datos de las papeletas para una elección.

Aquí adelante se encuentra una tabla con t y s de 2048 bits (el tamaño de clave mínimo considerado seguro hoy en día), mostrando como se ve afectado el tamaño de la elección por el número de opciones y el número de votantes.

Número de opciones	Número de votantes	Tamaño en bits máximo	Tamaño máximo
2	20	860160	107.52 KB
2	1000	43008000	5.37 MB
2	1000000	43008000000	0.57 GB
2	200000000	8601600000000	1.08 TB
10	20	4136960	517.2 KB
10	1000	206848000	25.856 MB
10	1000000	206848000000	26.856 GB
10	200000000	41369600000000	5.1712 TB

Tabla 6.1: Tamaño del voto según opciones y número de votantes.

Para efectuar correctamente los mecanismos de verificación de la blockchain es necesario que todas y todos los participantes tengan una copia de la cadena. Podemos observar aquí que no hay problema con pequeñas elecciones con un número reducido de votantes, pero que al acercarnos al tamaño de las elecciones nacionales, se vuelve imposible para la persona común participar en la verificación de la *blockchain*. Como último, es importante comentar que utilizar la variante de curvas elípticas de ElGamal ahorraría en el tamaño de la clave.

Por otro lado, hay que tomar en cuenta el proceso de escrutar los votos. Para realizar el cálculo del logaritmo discreto de la suma resultante de los votos es necesario generar la *lookup table* del tamaño de los votantes donde cada entrada en la tabla implica operaciones matemáticas entre números muy grandes. Para esto se recomienda la implementación de una librería a bajo nivel, ya que los resultados de tiempo para Python no son particularmente rápidos, llegando al orden de horas cuando las elecciones superan el millón de votantes usando claves de 2048 bits.

6.2 Coercibilidad del voto

Una característica necesaria en los sistemas de votación reales es la no coercibilidad. Se debe garantizar que el/la votante pueda siempre actuar sin coerción al momento de votar: debe estar libre de cualquier amenaza o incentivo de votar por quien no quiera. Estos casos de coerción incluyen

compra y venta de votos, votos bajo soborno o chantaje e incluso votos bajo amenaza de violencia.

El sistema propuesto no es libre de coerción de por sí, para eso tendría que haber unas garantías sobre canales físicos que no están contempladas en el marco de este trabajo de fin de grado.

En [16], basado en estudios sobre *multi-party computation*, se contempla un sistema que es incoercible basado únicamente en sus características criptográficas, sin necesidad de otras garantías. Dicho sistema incoercible es posible si se tiene un sistema de cifrado que fuese (a) de cifrado negable (*deniable encryption*), (b) completamente homomórfico y (c) apropiado para técnicas criptográficas de umbral. Actualmente no existe un criptosistema que posea estas características y sea eficiente.

6.3 Protocolos de consenso alternativos

Es importante también mencionar alternativas a *proof-of-work* y, en general, alternativas a algoritmos basados en uso de tiempo de procesador. Esto es debido a que dichos algoritmos poseen dos grandes problemas: privilegian a personas o grupos de personas con mayor acceso a poder computacional –como pueden ser bancos e instituciones privadas–, y requieren una cantidad ingente de recursos. Esto junto con otros problemas de menor calado: son poco viables en dispositivos móviles e IoT, requieren descargas de cantidades ingentes de información, etc. Es necesario presentar respuestas a los dos grandes problemas que enfrenta para desarrollar usos de la *blockchain* que sean coherentes con las necesidades que resuelven.

En primer lugar, los algoritmos que dependen sobre el poder computacional de los participantes en la seguridad de la *blockchain* dan más poder a quienes puedan pagar por él. En un sistema financiero como es el caso de las criptomonedas, esto es de esperarse y es parte del funcionamiento normal del sistema: las entidades con mayor poder económico van a ser las más interesadas en mantener el sistema seguro para sí mismas. Sin embargo, cuando la misma tecnología es utilizada para fines no económicos se entra en un conflicto de intereses y surgen problemas de seguridad, ya que la integridad de la información puede ser literalmente comprada.

Por otro lado, los algoritmos que sustentan la seguridad de la mayoría de

las criptomonedas actuales utilizan recursos energéticos de una manera muy poco eficiente. La carga energética diaria necesaria para asegurar la integridad de información para las criptomonedas de mayor uso –como son Bitcoin y Ethereum– es del orden del uso energético diario de países enteros. El uso correcto de la energía es cada vez más importante, tanto por la carga económica que conlleva, como por el impacto climático que produce, sobre todo mientras no hayamos completamente migrado nuestra producción energética a formas renovables.

Frente a esta doble problemática se presenta una solución: utilizar protocolos de consenso que no se basan en poder computacional sino utilizan protocolos de tolerancia de fallas bizantinas (*Byzantine Fault Tolerance*, BFT). El problema principal que afrontan estos sistemas alternativos en la actualidad es la escalabilidad, pero es un campo abierto y activamente en desarrollo. Un protocolo posible de usar es el descrito en [19]. Otros sistemas utilizando *blockchain* (o *distributed ledgers*, como prefieren llamarlo) ya han optado por utilizar sistemas similares, tales como [20, 21, 22, 23].

Capítulo 7
Conclusiones y líneas futuras

El desarrollo del sistema ha tenido que ser acotado fuertemente, ya que desarrollar un sistema con las características necesarias para el despliegue en situaciones reales se sale del alcance de un trabajo de fin de grado. Sin embargo, se ha desarrollado un sistema que implementa las últimas tecnologías para demostrar la posibilidad de herramientas democráticas alternativas. Se ha hecho un estudio de las consideraciones prácticas del sistema desarrollado y se ha demostrado su viabilidad para casos específicos.

La contribución de este trabajo ha sido adaptar sistemas de votación pensados como centralizados a sistemas descentralizados utilizando las tecnologías más apropiadas para ello. Se ha partido de Helios como base, aplicando las mejoras que se han planteado posteriores a su desarrollo y se ha buscado una manera de desplegar el sistema sobre una arquitectura descentralizada con el objetivo de asegurar las características planteadas en un principio.

En un futuro, se pueden implementar una serie de mejoras que han sido discutidas a lo largo de este documento, a resaltar:

- Cambiar a utilizar firmas y claves basados en curvas elípticas permitiría ahorrar en espacio en disco y en poder computacional.
- Como fue discutido previamente, los sistemas que operan con proof-of-work son costosos y no son buenos para el medio ambiente. Más investigación sobre la aplicación de algoritmos de tolerancia de fallas bizantinas en blockchains podría resultar en uso más extendido –y más inteligente– de esta tecnología.
- El problema de la coercibilidad sigue siendo un problema en la

votación electrónica y lo seguirá siendo mientras no se diseñen medidas que puedan atacar específicamente este problema.

Por último es importante repetir que, mientras se puedan crear sistemas de votación electrónica sumamente sofisticados y seguros, todo dependerá de la adaptación del público a estos sistemas para darles un uso correcto.

Capítulo 8
Summary and Conclusions

The project's scope had to be cut extensively in order to fit within the context of an undergraduate's final work. However, a system has been designed and implemented which shows the viability of alternative democratic tools. The consequences of the design choices have been weighed and commented as to provide an assurance of the correct working of the system under determined conditions. Our contribution lies in adapting originally centralized voting mechanisms to decentralized solutions utilizing blockchain technology.

This project's contribution comes from the adaptation of originally centralized voting systems to decentralized voting systems utilizing the most correct technologies available. We have started with Helios as a base to improve upon, along with the amendments and recommendations that have been brought up after its development. A way to deploy centralized systems on decentralized architectures has been proposed and implemented as to ensure the system fullfils the needed characteristics laid out in its design.

As for the future of this project, there still remains much work to be done to implement this system in real case scenarios. Some recommendations follow.

- Switching the chosen encryption cryptosystems for elliptical-curve-based systems would allow to save on disk space and computation time.
- As discussed previously, proof-of-work systems are expensive and not ecologically friendly. Further research into the application of byzantine fault tolerance algorithms in blockchains could result in a wider, more intelligent, use of this technology.

- Non coercibility still remains an issue in electronic voting and will continue to be until the correct measures are implemented.

Lastly, it's important to reiterate that while it's possible to create highly sophisticated and secure voting systems, the correct use of these systems depends on the public's adaptation to them.

Capítulo 9
Presupuesto

En el siguiente capítulo se desglosa el trabajo del trabajo de fin de grado en tareas, acompañadas con sus costes y tiempos.

9.1 Desglose del presupuesto

Tipos	**Descripción**	**Costo**	**Justificación del costo**
Programadores	Son necesarios programadores para implementar el código. Un sólo programador puede realizar toda la implementación la aplicación.	~20.800,00 €	Un programador pagado a 10€ la hora, trabajando 5 días a la semana, 8 horas al día.
Servidor	Un servidor para realizar pruebas es recomendable.	~24,63€ ($28,51)	El precio promedio de alquiler de un servicio de servidor de prestaciones bajas ronda los $0,0132 la hora. El servidor se puede alquilar durante los tres últimos meses de desarrollo, donde es particularmente necesario su uso.

Tabla 9.1: Presupuesto

Bibliografía

1. Potential and challenges of e-voting in the European Union http://www.europarl.europa.eu/RegData/etudes/STUD/2016/556948/IPOL_STU(2016)556948_EN.pdf
2. Boucher, P. How Blockchain technology could change our lives. European Parliamentary Research Service. http://www.europarl.europa.eu/RegData/etudes/IDAN/2017/581948/EPRS_IDA(2017)581948_EN.pdf
3. Helios: Web-based Open-Audit Voting . Ben Adida. Harvard University. https://www.usenix.org/legacy/events/sec08/tech/full_papers/adida/adida.pdf
4. Electing a University President using Open-Audit Voting: Analysis of real-world use of Helios. https://csrc.nist.gov/csrc/media/events/end-to-end-voting-system-workshop/documents/papers/demarneffe_papere2e.pdf
5. How not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios https://eprint.iacr.org/2016/771.pdf
6. Cryptographic Voting — A Gentle Introduction https://eprint.iacr.org/2016/765.pdf
7. Blockchain Voting Used By Danish Political Party. CCN. https://web.archive.org/web/20180525081636/https://www.ccn.com/blockchain-voting-used-by-danish-political-party/
8. Why ripples from this Estonian blockchain experiment may be felt around the world. zdnet https://web.archive.org/web/20180525082154/https://www.zdnet.com/article/why-ripples-from-this-estonian-blockchain-experiment-may-be-felt-around-the-world/
9. Agora Whitepaper version 0.2 https://agora.vote/Agora_Whitepaper.pdf?v=0.3
10. Hold Up: What Actually Happened in Sierra Leone's "Blockchain" Election?. Futurism. https://futurism.com/sierra-leone-election-blockchain-agora/
11. Internet Voting Using Zcash https://eprint.iacr.org/2017/585.pdf
12. Towards Secure E-Voting Using Ethereum Blockchain https://www.researchgate.net/publication/323318041_Towards_Secure_E-Voting_Using_Ethereum_Blockchain
13. The Council of Europe's Standards on e-Voting Pnyx Compliance with the Council of Europe's Security & Audit Standards on e-Voting December 2004 Scytl S.A., Barcelona, Spain. https://www.scytl.com/wp-content/uploads/2013/05/Pnyx_Compliance_with_CoE_Standards.pdf
14. Legal, operational and technical standards for e-voting https://www.coe.int/t/dgap/goodgovernance/activities/e-voting/key_documents/rec(2004)11_eng_evoting_and_expl_memo_en.pdf
15. Performance Analysis and Application of Mobile Blockchain https://arxiv.org/pdf/1712.03659.pdf

16. A Secure and Optimally Efficient Multi-Authority Election Scheme https://www.win.tue.nl/~berry/papers/euro97.pdf

17. PEP 8 -- Style Guide for Python Code https://www.python.org/dev/peps/pep-0008/

18. PEP 257 -- Docstring Conventions https://www.python.org/dev/peps/pep-0257/

19. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus https://www.stellar.org/papers/stellar-consensus-protocol.pdf

20. Transaction Flow. Hyperledger Fabric Wiki. https://hyperledger-fabric.readthedocs.io/en/release-1.1/txflow.html

21. The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. Marko Vukolić, IBM Research – Zurich. http://vukolic.com/iNetSec_2015.pdf

22. Algorand: Scaling Byzantine Agreements for Cryptocurrencies https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf

23. Proof of Work Without All the Work: Computationally Efficient Attack-Resistant Systems https://arxiv.org/pdf/1708.01285v3.pdf