UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Implementación distribuida de un protocolo de voto electrónico

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Rosello Ibañez, Oscar

Tutor/a: López Rodríguez, Damián

Cotutor/a: Muñoz Escoí, Francisco Daniel

Director/a Experimental: LARRIBA FLOR, ANTONIO MANUEL

CURSO ACADÉMICO: 2022/2023

Resumen

Los sistemas de voto electrónico son cada día más comunes, facilitando a los electores el proceso y/o permitiendo la reducción logística en determinados escenarios. En este proyecto se considera un protocolo de voto que garantiza las propiedades de seguridad e integridad deseables en estos sistemas sin recurrir al cifrado de la información. Este sistema, además, permite la verificabilidad del voto por parte de los electores, lo que permite que se conviertan en auditores del proceso. El proyecto plantea la implementación de un prototipo de este protocolo de voto electrónico. El desarrollo supone la implementación distribuida de los distintos roles implicados: autoridad de identificación encargada de validar el acceso, interventor de cada una de las partes implicadas y electores. Esta implementación distribuida admite un escalado adecuado de la aplicación en un eventual despliegue.

Palabras clave: Voto electrónico; verificabilidad; implementación distribuida; despliegue.

Resum

Els sistemes de vot electrònic són cada dia més comuns, facilitant als electors el procés i/o permetent la reducció logística en determinats escenaris. En aquest projecte es considera un protocol de vot que garanteix les propietats de seguretat i integritat desitjables en aquests sistemes sense recórrer al xifrat de la informació. Aquest sistema, a més, permet la verificació del vot per part dels electors, el que permet que es converteixin en auditors del procés. El projecte planteja la implementació d'un prototip d'aquest protocol de vot electrònic. El desenvolupament suposa la implementació distribuïda dels diferents rols implicats: autoritat d'identificació encarregada de validar l'accés, interventor de cada una de les parts implicades i electors. Aquesta implementació distribuïda admet un escalat adequat de l'aplicació en un eventual desplegament.

Paraules clau: Vot electrònic; verificabilitat; implementació distribuida; desplegament.

Abstract

Electronic voting systems are becoming more common every day, making the process easier for voters and/or allowing for logistical reductions in certain scenarios. This project considers a voting protocol that guarantees desirable security and integrity properties in these systems without resorting to information encryption. Additionally, this system allows for vote verifiability by voters, enabling them to become auditors of the process. The project proposes the implementation of a prototype of this electronic voting protocol. The development involves the distributed implementation of the different roles involved: the identification authority responsible for validating access, the intervenor for each of the parties involved, and the voters. This distributed implementation allows for appropriate scaling of the application in a potential deployment.

Keywords : Electronic voting, verifiability, distributed implementation, deployment.

Tabla de contenidos

1. Introducción

El voto electrónico introduce un nuevo conjunto de propiedades criptográficas para proporcionar confianza a los electores, tales como: verificabilidad universal, precisión o privacidad matemáticamente garantizada, que no están disponibles en el voto tradicional. Además, permite el voto remoto y el acceso desde múltiples dispositivos. Sin embargo, el voto electrónico todavía no logra ganarse la confianza de los electores y fomentar su participación. La mayoría de las personas se muestran reacias a confiar una parte crítica de la democracia a un sistema que no comprenden completamente.

A pesar de superar en cuanto a seguridad y eficiencia a los esquemas de votación tradicionales, la votación electrónica no está tan expandida y esto puede deberse a la desconfianza ante los sistemas de votación por diferentes motivos. Los sistemas pueden tener fallos de seguridad o pueden ser alterados por atacantes si poseen vulnerabilidades y no están implementados correctamente. En 2004 ocurrió el famoso caso de Diebold [1] en el que se filtró el código fuente de sus sistemas de votación y en el que los expertos en ciberseguridad encontraron graves fallos de seguridad. Otro motivo es la falta de transparencia existente en los principales sistemas de votación electrónica, el usuario puede pensar que su voto no ha sido registrado o que el sistema está amañado para beneficio de alguien. Esto se debe a que el elector no recibe nada que le asegure que el voto ha sido registrado con el valor correcto. Este tema ha sido muy controvertido en Estados Unidos, donde ha habido múltiples acusaciones de amaños en las votaciones y donde la votación se realiza en unas máquinas ubicadas en los colegios electorales. En las máquinas, el elector marca en la pantalla a quien desea votar, siendo un mensaje de información lo único que ve tras la elección. Por último, otro problema que se presenta es que, si la votación se realiza desde un dispositivo del usuario, este debe estar limpio de *malware* puesto que esto podría alterar el proceso de votación y modificar el voto del usuario a favor del atacante.

En este trabajo se analizará, implementará y realizará un pequeño despliegue del protocolo de votación electrónica propuesto por los investigadores de la "Universitat Politècnica de València" Damián López y Antonio Larriba en su artículo *SUVS: Secure Unencrypted Voting Scheme* [2]. En su estudio, introducen un esquema de votación electrónica que no hace uso de criptografía sin comprometer la seguridad. Inspirado por el trabajo de Shamir [3], el voto se concibe como piezas fragmentadas de información que no revelan ninguna información sobre el voto original de forma separada pero que al juntarlas permiten la recuperación del voto. Esta implementación pretende ser escalable e incluye un boletín público en el que los usuarios podrán confirmar que su voto ha sido correctamente procesado.

1.1 Motivación

La votación electrónica es un tema de actualidad y puede ser interesante investigar sobre sus ventajas e inconvenientes en comparación con el sistema tradicional de votación. Intentar eliminar la desconfianza ante la votación electrónica presenta grandes ventajas puesto que estos mejoran en cuanto a eficiencia, costo, transparencia y seguridad a los sistemas tradicionales, eliminando la necesidad de personal humano para el control y conteo de votos, la posibilidad de error humano y, por ende, dificultando la posibilidad de fraude electoral. Además de todo lo mencionado anteriormente, la votación electrónica puede facilitar la vida de la gente dado que aumenta la accesibilidad al voto. Permite realizar el proceso de votación desde el mismo hogar con cualquier dispositivo sin necesidad de acudir a una mesa electoral y esto, de la misma manera que al total de los votantes, puede resultar de gran ayuda para la gente que sufre de discapacidades o limitaciones físicas.

En cuanto al protocolo propuesto, hasta donde sabemos, es el primer protocolo de votación electrónica que no hace uso de criptografía para ocultar el voto sin afectar a la seguridad de la votación. Otra ventaja del esquema es que, debido a la flexibilidad en cuanto a la codificación del voto se refiere, es compatible con cualquier tipo de elección existente. Además, el protocolo se muestra altamente eficiente y la seguridad de este no se ve afectada por el hecho de no encriptar los votos. El esquema escala linealmente con el número de votos procesados por lo que permite plantearlo en situaciones en las que el número de votos vaya a ser elevado. Por último, el esquema presenta un boletín público que los usuarios pueden consultar para confirmar que su voto ha sido correctamente procesado, ofreciendo transparencia para los votantes y aumentando su confianza.

Por estos motivos, considero que la implementación de un sistema de votación electrónica puede aportar valor en cuanto a mejora en la sociedad y en cuanto a valor teórico se refiere. El análisis, implementación y testeo de estos sistemas provoca un avance respecto al estudio teórico para la mejora de estos esquemas, produciendo sistemas cada vez más eficientes y seguros que a su vez ayudan a aumentar la confianza de la gente en estos. Todo esto nos acerca cada vez más a un entorno en el que la votación electrónica esté a la orden del día y sea una herramienta que sirva para facilitar la vida de la gente.

1.2 Objetivos

El objetivo principal de este trabajo es implementar un prototipo basado en el protocolo de votación SUVS. Esta implementación debería proporcionar una base para estudiar las posibilidades de escalado de la solución, las condiciones a tener presentes en un despliegue, así como ayudar en la detección de vulnerabilidades a tratar convenientemente en una implementación futura. Para ello se pretende conseguir lo siguiente:

- Realizar un análisis previo del esquema de votación electrónica en el que se expondrán los detalles del esquema y la complejidad y seguridad de este. (El objetivo de este punto es que los detalles sobre el protocolo queden claros).

- Implementar en Node.js el protocolo y los tres roles que forman el esquema de votación, así como la comunicación entre estos.

- Realizar un pequeño despliegue mediante Docker en el que se simule una situación real de votación.

- Realización de un conjunto de pruebas sobre el prototipo para comprobar que, efectivamente, el protocolo está correctamente implementado y es seguro y eficiente.

1.3 Estructura

El resto de la memoria está estructurada como sigue:

- En el capítulo 2 (**Estado del arte**) se repasa la literatura más relevante y las tecnologías utilizadas actualmente en lo que a votación electrónica se refiere.

- El capítulo 3 (**Análisis**) muestra la planificación y la gestión del proyecto e introduce y detalla los aspectos que forman el protocolo de votación electrónica, analiza su complejidad y su seguridad.

- En el capítulo 4 (I**mplementación**) se trata la implementación en Node.js del esquema y se explica cómo y por qué se ha implementado de la manera que se ha realizado.

- El capítulo 5 (**Despliegue**) explica el despliegue realizado en Docker mediante Docker-compose.

- En el capítulo 6 (**Pruebas**) se estudia el comportamiento del esquema y se presentan las diferentes pruebas realizadas.

- El capítulo 7 (**Conclusión y trabajo futuro**) muestra un conjunto de aspectos a tener en cuenta para el trabajo futuro y la mejora del proyecto y una breve conclusión sobre el proyecto.

2. Estado del arte

La votación electrónica es un área con una gran investigación a sus espaldas y muchos enfoques han abordado el problema desde diferentes ángulos. En esta sección se pretende exponer algunas de las aproximaciones más habituales en el diseño de protocolos de votación electrónica revisando brevemente contribuciones relevantes en cada una de ellas y las técnicas que utilizan.

2.1 Protocolos sin uso de criptografía

En Three Ballot de Rivest [4]. Aunque este protocolo no considera votos digitales ni la emisión remota de ellos, la propuesta de Rivest oculta los votos de los electores sin necesidad de utilizar ninguna criptografía. Rivest propone que la elección se base en boletas de papel. Estas boletas están formadas por tres secciones que el elector completa según un patrón para mostrar su aprobación o rechazo a los candidatos. Cuando el voto se completa correctamente, sus tres secciones se separan y se emiten de manera independiente. El elector recibe una copia de una de estas secciones como recibo, para garantizar que todos los votos se cuenten en el recuento final. Sin embargo, el recibo no puede convencer a nadie de la dirección del voto.

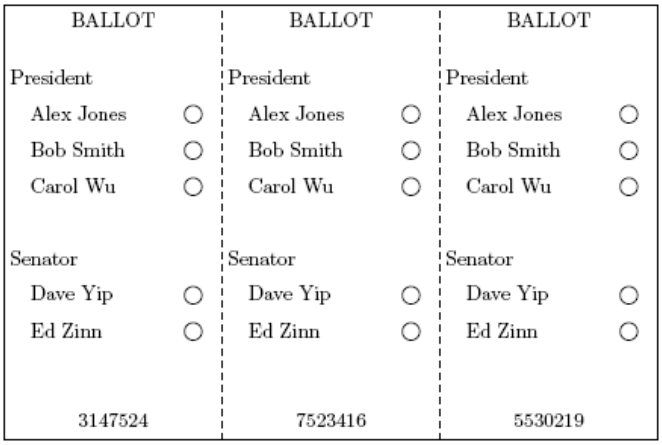

Figura 1. Ejemplo de papepeleta del protocolo Three Ballot. [4]

Este protocolo se amplía en Rivest y Smith [5] para hacerlo compatible con cualquier tipo de elección (por ejemplo, Borda, Ranking o Condorcet). Desafortunadamente, ambos enfoques sufren el llamado "ataque de patrón triple", donde un coercitivo puede comprar o influir en votos al requerir que los electores completen las boletas en ciertos patrones anómalos. Más tarde, el atacante puede buscar estos patrones en el boletín público que contiene el recuento final. Este ataque no tiene éxito si el patrón es simple, ya que es probable que cada patrón ocurra muchas veces. Sin embargo, esta suposición restringe los escenarios aplicables para estos protocolos.

2.2 Firma ciega

La firma ciega fue propuesta por Chaum [6] como un método para firmar un mensaje enmascarado sin la posibilidad de conocer el contenido del mismo.

En Li et al. [7], se presenta un sistema de votación que emplea firmas ciegas para certificar el voto. Los votantes reciben una papeleta en blanco de la entidad certificadora y proceden a usar la técnica de firma ciega para certificar el voto. Una vez certificado, se elimina la máscara, manteniendo la firma, y se procede a encriptar el voto para enviarlo a la autoridad de recuento de votos. La encriptación se usa para mantener los votos ocultos hasta el proceso de conteo. Una vez finalizado el proceso de votación, la autoridad certificadora desvela la clave secreta para que la autoridad de recuento desencripte, verifique y cuente los votos.

En Nguyen y Dang [8], los autores presentan otro esquema que utiliza firmas ciegas para la certificación de papeletas para garantizar la privacidad del elector. Los electores se registran utilizando su identificación personal para obtener un certificado digital. A continuación, para obtener la firma de la autoridad de privacidad, el elector oculta su identificación única mediante firmas ciegas, y la envía junto con el número de serie del certificado. Para evitar los ataques “man in the middle”, emplean triple DES para cifrar las comunicaciones sensibles. Una vez que el usuario obtiene la firma que le da derecho a votar, elabora una papeleta dinámica y la envía al centro de votación. La papeleta está encriptada y el elector debe presentar pruebas de equivalencia en texto plano para demostrar que las identificaciones únicas son válidas.

Otro protocolo que también emplea firmas ciegas para certificar las papeletas se describe en Larriba et al. [9]. En su trabajo, los autores se centran en proponer un esquema de votación eficiente que sólo requiere dos autoridades. En este protocolo el elector construye la papeleta de acuerdo con una estructura fija. Esta papeleta está firmada a ciegas por una autoridad que no puede desvelar el sentido del voto. A continuación, la papeleta firmada se envía a la autoridad que desempeña el papel de colegio electoral. A menos que ambas autoridades se confabulen, el método mantiene todas las propiedades deseadas.

Muchos otros esquemas de votación explotan las propiedades de las firmas ciegas, ya que proporcionan un método flexible para firmar los votos y evitar el doble voto sin comprometer la privacidad del elector [10] [11] [12].

2.3 Protocolos homomórficos

La criptografía homomórfica permite realizar operaciones con textos cifrados y obtener, tras el descifrado, los mismos resultados que obtendríamos si hubiésemos realizado estas mismas operaciones con texto plano. Esto permite trabajar con datos confidenciales sin necesidad de conocer el contenido real del mensaje cifrado. De hecho, las firmas ciegas se benefician de la criptografía homomórfica para conseguir su

objetivo. La mayoría de los protocolos que usan criptografía homomórfica trabajan con los votos encriptados y solo desencriptan el resultado final de la votación.

$$E(x)\ \boldsymbol{op_1}\ E(y) = E(x\ \boldsymbol{op_2}\ y), Donde\ \mathbf{op}\ es\ cualquier\ operación.$$

En Cramer et al. [13], el autor presenta un esquema de votación para elecciones Sí/No basado en propiedades homomórficas. Los votos, que se codifican como (1, −1), se cifran utilizando un criptosistema umbral de El Gamal [14]. Luego se agregan los votos encriptados, puesto que se pueden sumar como números enteros sin alterar el resultado de la operación dado a las propiedades homomórficas de ElGamal. Al final de la elección, las autoridades colaboran para recuperar la clave secreta y descifrar el resultado final.

En Yang et al. [15] [16], se introduce un esquema de votación homomórfica compatible con la votación por rango. La votación por rango requiere que los electores asignen un puntaje numérico a todos los candidatos para las elecciones de un solo escaño. El candidato con la puntuación más alta gana. En este sistema, los votos están estructurados y encriptados como elementos de una matriz en la que las filas representan a los candidatos y las columnas representan el puntaje asignado. Dado que los votos para los mismos candidatos se colocan en las mismas filas de la matriz, se pueden sumar para obtener los resultados finales por candidato. Al final de la elección, las autoridades colaboran para recuperar la clave secreta y descifrar la matriz final que aglutina todos los votos.

2.4 Firma en anillo

Las firmas en anillo son un tipo especial de firma digital que permite a cualquier usuario firmar como miembro de un colectivo en lugar de individualmente. El verificador puede comprobar que el usuario que quiere firmar pertenece a un grupo determinado, pero no puede identificar al usuario entre los miembros del grupo.

Tornos et al. [17] proponen un esquema de votación que proporciona ambigüedad del firmante mediante el uso de firmas de anillo. Se emplea una sola autoridad de registro para manejar la identificación adecuada de electores. Después del proceso de identificación, los electores usan firmas anulares para firmar sus votos en privado. Para evitar la doble votación, utilizan etiquetas de enlace en las firmas del anillo. Estas etiquetas no revelan información personal sobre el firmante, pero le impiden votar más de una vez usando diferentes anillos de usuarios.

En Chen et al. [18], se presenta un protocolo de votación con firmas personalizadas en anillo. Se propone un esquema de firma de anillo que solo puede ser verificado por algunas personas designadas. Estos verificadores no pueden convencer a un tercero de la integridad de una firma de anillo sin revelar su clave privada. Para la encriptación del voto emplean un esquema de umbral en el que las autoridades tienen que colaborar

para recuperar la clave secreta al final de la elección. Para evitar la doble votación, los electores deben vincular el voto con un compromiso de su llave privada. Este compromiso no revela ninguna información sobre la clave privada, pero evita que los electores malintencionados voten varias veces. Si en dos o más votos se detecta el mismo compromiso, solo se considerará el que tenga la última marca de tiempo.

2.5 Blockchain

Blockchain proporciona una tecnología descentralizada para almacenar y compartir información. Múltiples esquemas de votación electrónica han utilizado blockchain para llevar a cabo la elección ya que proporcionan un libro público distribuido que puede ser consultado por cualquier persona. Si bien blockchain no es un esquema criptográfico en sí mismo, y todos estos sistemas usan otras primitivas criptográficas para lograr la privacidad, no deja de ser un factor diferenciador que merece una distinción al estudiar estos protocolos.

En Yang et al. [19], los autores proponen un protocolo de votación para la votación por rango que utiliza blockchain para estructurar el proceso electoral. En este esquema, cada candidato recibe un puntaje (por ejemplo: token en la cadena) de los electores y el candidato con el puntaje más alto gana la elección. Para preservar la privacidad de los electores, emplean un sistema de encriptación basado en El Gamal y cifrado basado en grupos. Se aprovechan las propiedades homomórficas de El Gamal para calcular la cuenta final sin comprometer a los electores individuales.

En Gao et al. [20], los autores presentan un protocolo de voto electrónico basado en la tecnología blockchain, que también proporciona propiedades postcuánticas. Para lograrlo, basan su método en un problema NP-completo [21] en lugar de utilizar criptografía tradicional de clave pública. Su protocolo posee una función de auditoría que permite detectar votantes fraudulentos respetando su privacidad.

En Larriba et al. [22], los autores proponen un esquema basado en blockchain que introduce a los partidos tradicionales dentro del proceso electoral para aumentar la confianza en el sistema. Para proteger la privacidad de los electores, emplean firmas circulares y, para evitar la doble votación, emplean imágenes clave. Las imágenes clave actúan como recibos de firmas circulares que evitan que el elector malintencionado cree varias firmas sin comprometer su identidad.

2.6 Técnicas usadas en SUVS

El SUVS aprovecha la idea de dividir el voto en distintas partes para evitar el tener que encriptar los votos. Además, se usa la técnica de firma ciega y, por lo tanto, propiedades homomórficas para que solo pueda votar la gente que esté censada y que no pueda hacerlo dos veces. En el esquema de votación del trabajo, no se emplea firmas en anillo ni blockchain. Sin embargo, el boletín público que se utiliza para comunicar los resultados de las elecciones podría implementarse utilizando tecnología blockchain.

3. Análisis

Antes de empezar con la presentación del esquema, se introducirán las tareas realizadas en el proyecto junto con su coste y las primitivas criptográficas fundamentales que se emplean en el protocolo.

3.1 Planificación

Este proyecto consiste en 5 fases bien marcadas las cuales son: estudio del esquema, implementación, despliegue, pruebas y realización de la memoria. El diagrama de Gantt que representa el trabajo realizado en estas fases es el que sigue:

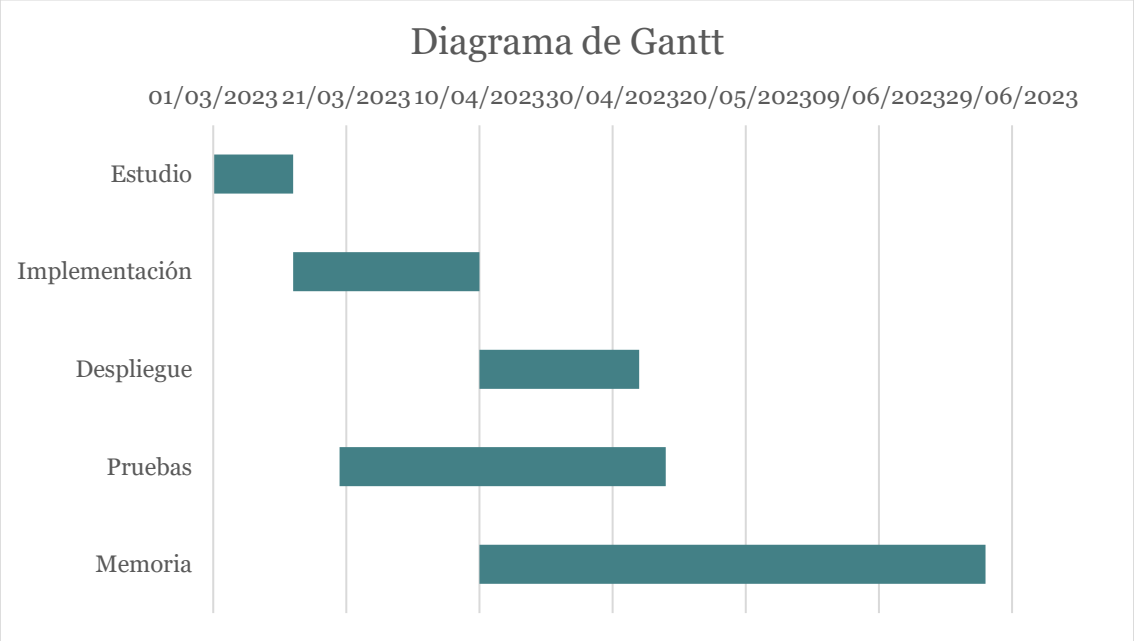

Figura 2. Diagrama de Gantt del proyecto

Procederemos pues a analizar el trabajo realizado y la duración en cada una de las fases.

3.1.1 Estudio

Durante esta fase, que tiene una duración de 12 días, se procede a estudiar el artículo en el que se basa este proyecto además de otros artículos relacionados con este para entender el protocolo y poder empezar con la implementación.

Podemos observar que todas las tareas que siguen dependen de esta puesto que hasta que no se haya realizado el análisis previo de todos los conocimientos requeridos no se empezará a desarrollar el proyecto.

3.1.2 Implementación

Una vez terminada la fase de estudio, se procede a la fase de implementación la cual tiene una duración de 28 días. En esta fase se escribe el código fuente de los diferentes roles que conforman el esquema. Para esto, se utiliza Node.js y Visual Studio Code como IDE.

De esta tarea dependen directamente el despliegue y la realización de la memoria. El despliegue no puede ser realizado hasta que el código fuente no funcione correctamente y para poder empezar la memoria, sería recomendable haber realizado un mínimo del proyecto.

3.1.3 Despliegue

Terminada ya la implementación, se empieza con el despliegue del esquema. Esta fase tiene una duración de 24 días. Para esta fase se utiliza la herramienta Docker que nos servirá para crear y desplegar todos los contenedores.

Las dependencias de esta fase son: la fase de pruebas, que no se puede realizar hasta que no esté correctamente desplegado, y la fase de la memoria a partir del punto 5.

3.1.4 Pruebas

Las pruebas se realizan paralelamente a la implementación y el despliegue. Esta tarea tiene una duración de 49 días y en ella se comprueba que todo funciona correctamente y se corrige todo aquello que estuviese causando errores. Además de esto se genera un caso de uso para comprobar como se comporta el esquema ya implementado y desplegado.

Esta fase depende de todas las demás excepto de la memoria y lo único que depende de ella es la parte final de la memoria.

3.1.5 Memoria

Paralelamente al inicio del despliegue se empieza con la redacción de la memoria. Esta es la fase más larga pues tiene una duración de 76 días. Se realiza al mismo tiempo que las fases finales y depende de estas puesto que hasta que no se terminen no se puede empezar a redactar sobre ellas.

Cabe recalcar que de esta fase no depende ninguna tarea puesto que podría haberse realizado al final del proyecto y no hubiese generado ningún conflicto.

3.2 Conocimientos previos

3.2.1 Esquema de Intercambio de Secretos de Shamir

El enfoque reconstructivo se basa en el trabajo de intercambio de secretos (j, d) de Shamir [3]. Este esquema permite compartir un secreto C entre j jugadores, de tal forma que cualquier subconjunto d + 1 de jugadores puede recuperar C. El secreto se codifica como el término independiente de una función polinómica q(x) y la información distribuida entre los diferentes jugadores son puntos pertenecientes a la función q(x). Así pues, cualquier subconjunto de jugadores d + 1 puede interpolar su conjunto de puntos para recuperar el secreto original C. Las operaciones de evaluación de polinomios se llevan a cabo bajo aritmética módulo p (siendo p un número primo) como se muestra a continuación:

$$q(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x^1 + C \ (mod\ p)$$

Dado un conjunto suficiente de puntos (xi, yi), para interpolar el polinomio q(x), se aplica el método de interpolación escogido para reconstruir la función. En el caso del SUVS se escoge el método de Lagrange:

$$q(x) = \sum_{i=0}^{j} y_i l_i(x),$$

donde:

$$l_i(x) = \prod_{\substack{0 \le k \le j \\ k \ne i}} \frac{x - x_k}{x_i - x_k}$$

Nótese que el esquema de intercambio de secretos de Shamir requiere del uso de aritmética modular, sin embargo, no implica uso de criptografía.

3.2.2 Seguridad Perfecta

La noción de seguridad perfecta se deriva directamente de la teoría de la información [23]. Implica que la probabilidad a priori de un mensaje dado m, en el espacio de posibles mensajes M, es igual a la probabilidad a posteriori del mensaje dado el criptograma c, en el espacio de posibles criptogramas C.

$$P(M = m) \equiv P(M = m|C = c)$$

Así pues, bajo esta aproximación, los textos cifrados no revelan información sobre el mensaje. Todos los mensajes son equiprobables para un texto cifrado dado, lo que hace que el esquema sea seguro ya que el atacante no tiene método para obtener información adicional, incluso con textos cifrados seleccionados.

3.3 Secure Unencrypted Voting Scheme

El protocolo de votación SUVS emplea el esquema de intercambio de secretos de Shamir para dividir el voto en diferentes partes que por separado no revelan información, pero cuando son combinadas revelan el voto.

El sistema consiste en tres diferentes roles: el elector, la autoridad de identificación y los interventores. El elector se encarga de construir su voto, la autoridad de identificación tiene el propósito de validar el voto de los electores si estos están en el censo y no han votado ya y, por último, los interventores que se representan a sí mismos o a su partido como una opción en las elecciones y que son responsables de comprobar la validez del voto, recuperarlo y proceder al conteo. Los tres roles emplean un Boletín Público para comunicar la información relacionada a la votación.

El SUVS está formado por cinco fases secuenciales: configuración del sistema, la construcción del voto, la validación del voto, la emisión de los votos y la fase de escrutinio. A lo largo de estas fases, el elector genera un polinomio privado que actúa como su propio voto. El polinomio permite al elector ocultar su voto como un conjunto de puntos y emitirlo en la fase correspondiente. Se aprovechan las propiedades de la interpolación polinomial, que hacen que la recuperación del voto sea imposible si no se conocen todos los puntos. En la última fase, las partes colaboran para recuperar el polinomio secreto, y su voto asociado, a partir de los puntos recibidos. A continuación, se describirán al detalle todas las fases que conforman el protocolo.

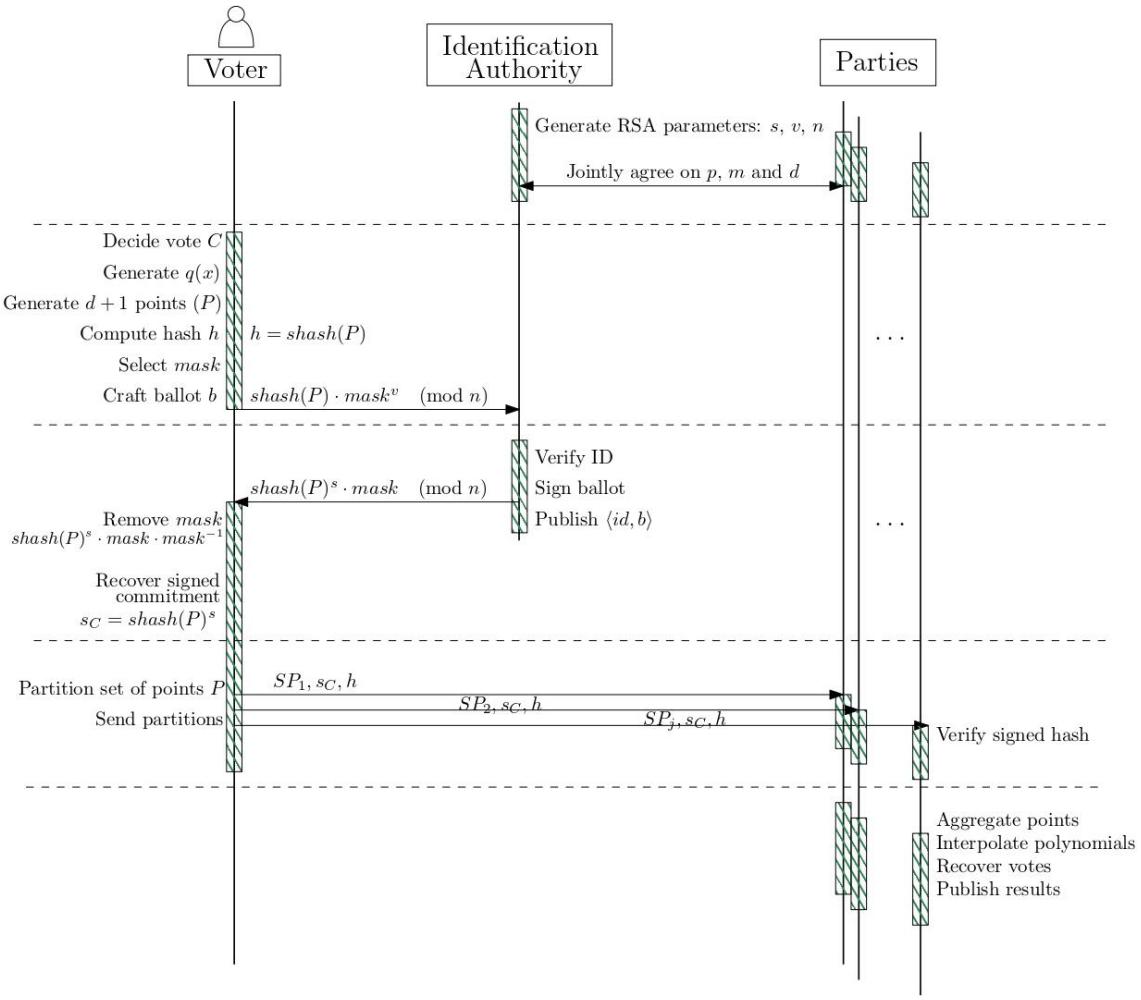

Figura 3. Esquema gráfico del protocolo. [2]

3.3.1 Configuración del sistema

Previo al proceso de elección, se requiere disponer y configurar los métodos que van a utilizarse para firmar las papeletas de los electores y para la construcción del voto.

La autoridad de identificación emplea firmas ciegas para evitar la votación de electores no censados y la doble votación sin comprometer la privacidad del voto. En el desarrollo del esquema se considera el uso del esquema criptográfico RSA (Rivest-Shamir-Adleman) para la implementación de firmas ciegas debido a sus propiedades homomórficas bajo la exponenciación modular.

La autoridad de identificación también establece la función hash a utilizar, el grado d de la función polinómica, el máximo número de puntos que un elector puede generar l, siendo $l > d$, así como el primo p que será el módulo sobre el que realizar las operaciones modulares. Nótese que d debe ser, al menos, igual a $j - 1$, siendo j el número de partidos involucrados en la votación forzando la colaboración de todos los

partidos para la recuperación de los votos. Aparte de esta consideración, aumentar d no proporciona mayor seguridad.

Se debe tener en cuenta que l se incluye como medida de seguridad para evitar que los usuarios generen demasiados puntos. Debido a que el grado d del polinomio es público, esto podría provocar que un pequeño conjunto de partidos maliciosos se alíe para recuperar los votos antes de la fase de recuento.

3.3.2 Construcción del voto

Una vez el elector ha decidido su voto, lo codifica como un entero C. Entonces, procede a generar un polinomio de grado d en el que C es el término independiente. No es necesario que los coeficientes sean no nulos, pero sí deben ser menores que p. Ejemplo:

$$q(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x^1 + C \ (mod\ p)$$

Una vez construida la función, el elector recopila una serie de puntos pertenecientes a q(x) que va desde un mínimo de $d + 1$ puntos hasta un máximo de l. Sin pérdida de generalidad, asumimos en el resto del trabajo que el usuario generará un número de puntos igual a $d + 1$. Así pues, se ordenará el conjunto de puntos basándose en la primera coordenada para transformar el conjunto en una secuencia de puntos.

$$P = \langle (x_1, q(x_1)), (x_2, q(x_2)), \dots, (x_{d+1}, q(x_{d+1})) \rangle$$

Cualquiera que conozca P puede interpolar el polinomio original q(x), y por lo tanto recuperar el voto.

El conjunto es, en realidad, la papeleta a emitir, que se dividirá en partes para enviar a los partidos. Para permitir la reconstrucción de la papeleta dividida, el elector digiere el conjunto ordenado de puntos P utilizando la función hash seleccionada en la fase de configuración del sistema. Es importante que los puntos se ordenen antes de aplicarle la función hash puesto que así generaremos salidas más consistentes, así pues, denotaremos como función shash aquella que ordena los puntos antes de aplicarle la función hash mencionada. La salida de la función shash nos vale como compromiso de los puntos el cual asegura que el conjunto de puntos P no ha sido alterado y demuestra la validez del voto cuando se firma.

3.3.3 Certificación del voto

Para evitar que un usuario no censado o que uno que ya haya votado envíen su papeleta y se cuente, el elector envía su voto a la autoridad de identificación para que lo certifique. A fin de eludir el hecho de que en este proceso la autoridad de identificación pueda asociar el voto con un elector se utilizan firmas ciegas y el votante enmascara el voto antes de enviarlo a la autoridad.

Para realizar este proceso, el elector genera una máscara como un entero invertible módulo *p*, considera la clave pública de verificación *v* y construye su papeleta *b* como se muestra a continuación:

$$b = shash(P) \cdot mask^{v} \ mod \ p$$

El usuario envía la papeleta junto a su identificación a la autoridad de identificación y esta comprueba si la identificación es correcta y si está en el censo de votantes. Si todo es correcto procede a firmar el voto como sigue:

$$b \equiv (shash(P) \cdot mask^{v})^{s} \ (mod \ p)$$

$$b \equiv shash(P)^{s} \cdot mask^{vs} \ (mod \ p)$$

$$b \equiv shash(P)^{s} \ \cdot mask \ (mod \ p)$$

Téngase en cuenta que, a menos que la autoridad de identificación conozca la máscara, esta nunca puede ser consciente de qué mensaje está realmente certificando. Luego del proceso de firma, la papeleta firmada es devuelta al elector a través de un canal seguro. La autoridad de identificación también publica en el PBB una tupla de la forma $\langle id, \ b = (shash(P) \cdot mask^{v})^{s} \rangle$. El propósito de esta publicación es doble: primero, le permite al elector verificar que su papeleta fue recibida según lo previsto. En segundo lugar, demuestra que cada papeleta provenga de una identidad válida del censo público.

El elector recibe la boleta firmada y procede a recuperar el compromiso firmado que certificará su voto. Tenga en cuenta que el elector es el único que conoce la máscara y su inversa. Así, el elector obtiene el compromiso firmado como se indica en la ecuación:

$$b^{s} \cdot mask^{-1} \equiv shash(P)^{s} \cdot mask \cdot mask^{-1} \equiv shash(P)^{s} \ (mod \ p)$$

Al seguir estos pasos, el elector obtiene el compromiso certificado (firmado) que se utilizará en las próximas fases. Tenga en cuenta que, a pesar de requerir su identidad para firmar la papeleta, la autoridad de identificación no tiene medios para vincular el compromiso con el elector. También considere que el elector puede verificar si la

papeleta firmada fue manipulada durante el camino, porque puede verificar la integridad del compromiso firmado de forma independiente.

3.3.4 Emisión de los votos

En esta fase, el elector dispone de un conjunto P que puede utilizar para recuperar su voto y el compromiso firmado del conjunto P, que identifica su voto como válido.

Para finalmente emitir el voto, el elector envía una partición de P (partes de la papeleta) junto con el compromiso certificado a todos los partidos implicados en el cómputo electoral. Tenga en cuenta que una propiedad básica de la interpolación polinomial establece la imposibilidad de recuperar un polinomio de grado d con d o menos puntos de dicho polinomio. Esto permite enviar las diferentes partes a los diferentes partidos con certeza de que no se revela información del voto original y de que no será revelado a menos que todas las partes colaboren para hacerlo.

Así, teniendo en cuenta que k partidos están implícitos en la elección, el elector divide P en k subconjuntos no superpuestos SPi, de modo que P representa la combinación ordenada de los subconjuntos SPi.

$$P = \left\{ \bigcup_{\forall i \; 1 \leq i \leq k} SP_i \right\}$$

Cada una de las partes recibe una parte SPi de P junto con el hash y el certificado del hash. Tenga en cuenta que el hash certificado funciona como firma digital del hash, y que ambos son enviados necesariamente para comprobar su validez. También tenga en cuenta que el elector puede decidir libremente qué subconjunto se envía a cada partido.

También notamos que es posible reducir el número de acciones al apartar del proceso a aquellos partidos que no reciben parte del voto del elector. Esto no afectará a la validez del voto, pero sí la transparencia del proceso. Sin embargo, obligamos a que cada partido reciba una parte para aseguramos de que la votación requiera la colaboración de todas las partes para su recuperación. Por lo tanto, ningún subconjunto de partidos maliciosos puede recuperar el voto antes de la fase de escrutinio. Además, cuando se emite el voto, ninguna información personal va junto con las partes del voto por lo que los partidos no tienen medios para asociar las partes recibidas con el voto de un elector.

3.3.5 Escrutinio

Cuando terminan las elecciones, no se aceptan nuevos votos y los partidos pueden proceder a reconstruir los votos y empezar con el conteo de estos.

En primer lugar, los partidos consideran la certificación que acompaña a los puntos para encontrar el conjunto de puntos (cada uno de ellos recibido por un partido diferente) que permiten reconstruir cada papeleta P. Nótese que la secuencia original se puede obtener fácilmente, ordenando el conjunto P, y que es posible comprobar la validez de la certificación.

Se descartan los conjuntos de puntos P tal que su compromiso certificado no sea correcto, o que su cardinalidad supere el máximo definido (|P| > l). Una vez reconstruido P, los partidos pueden ahora, individualmente, usar los puntos en P para recuperar el polinomio original q(x) que contiene el voto como su término independiente C. Para recuperar un polinomio teniendo un conjunto de puntos:

$$P = \{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_j = (x_j, y_j)\},$$

se sugiere el uso de la interpolación de Lagrange:

$$q(x) = \sum_{i=0}^{j} y_i l_i(x),$$

donde:

$$l_i(x) = \prod_{\substack{0 \le k \le j \\ k \ne i}} \frac{x - x_k}{x_i - x_k}$$

Así pues, podemos recuperar el término independiente de esta forma:

$$C = q(0) \ (mod\ p)$$

Los partidos publican en el PBB, una tupla de 3 por voto que contiene: la certificación de la papeleta (es decir, el compromiso firmado); la papeleta misma (es decir, el conjunto de puntos P que ocultan la papeleta); y el voto reconstruido C. El cómputo final obtenido por cada partido también puede ser publicado. El PBB está a disposición de todos para comprobar que sus votos han sido contados según lo previsto, y para verificar la integridad del conteo final.

3.3.6 Complejidad y Seguridad

En esta sección se presentará el análisis realizado por Larriba y López [2] sobre la complejidad temporal del sistema de votación *SUVS*. Antonio Larriba y Damián López eligen la operación con bits como unidad en el análisis de complejidad temporal. Como de costumbre, n denota la entrada de los operandos y log n denota su número de bits. Además, diferencian entre la complejidad computacional relacionada con cada elector individual y la complejidad de todo el sistema para procesar todos los votos.

Para certificar y emitir un voto, el elector necesita llevar a cabo una serie de pasos: generar un polinomio, muestrear algunos puntos, calcular una función hash, seleccionar algunos subconjuntos, etc. Sin embargo, en el análisis de la complejidad temporal los autores no tienen en cuenta algunos de estos pasos por dos motivos principales: la mayoría de ellos pueden realizarse fuera de línea antes de que comience el proceso de elección y no son relevantes en términos de análisis de complejidad temporal porque otras operaciones dominan la complejidad global.

Los autores solo consideran la generación de máscaras y las operaciones de multiplicación y exponenciación. Así pues, la complejidad del elector para construir un voto es:

$$Generación\ de\ máscaras\ +\ Exponenciación\ modular\ +\ Multiplicación\ modular\ =$$

$$O(log^2 n)\ +\ O(log^2 n)\ +\ O(log^3 n)\ \approx\ O(log^3 n)$$

Para procesar un voto, SUVS debe aplicar una firma ciega a la papeleta y, al final de la elección, interpolar un polinomio a partir de un conjunto de puntos. Larriba y López no consideran la recopilación de las partes con el mismo compromiso certificado en el análisis de complejidad, las partes pueden ser indexadas y recopiladas en tiempo constante. Considerado esto, el coste del sistema para procesar los votos es:

$$Numero\ de\ votos\ *\ (Certificación\ de\ las\ papeletas\ +\ Interpolación\ de\ Lagrange)\ =$$

$$v(O(log^3 n) + O(log^4 n))\ \approx\ vO(log^4 n)$$

Como podemos observar, el coste del algoritmo para procesar los votos escala linealmente con el número de votos.

En cuanto a seguridad, el protocolo cumple con las propiedades deseadas de un protocolo electoral seguro. Se mantiene la privacidad del elector, pues ni la autoridad ni los interventores tienen forma de relacionar a un votante con el sentido de su voto. El votante puede verificar su voto en cualquier momento de la elección y puede comprobar que los demás no han sido alterados. Además, todo el mundo puede auditar las votaciones, comprobando que todos los votos han sido correctamente contados y que los votos incorrectos han sido desechados. El protocolo también asegura que solo los votantes censados puedan realizar el proceso de elección y que este solo pueda ser realizado una vez. Por último, en el artículo original también se demuestra que ningún elector, coalición de electores o coalición de partidos puede actuar de manera maliciosa para intentar romper con el proceso de votación.

Se recuerda al lector que todos estos enunciados están demostrados en el artículo científico SUVS: Secure Unencrypted Voting Scheme de Antonio Larriba y Damián López [2].

4. Implementación

En esta sección se abarcará la implementación del esquema anteriormente presentado. Para implementar el esquema se usará el entorno de ejecución de JavaScript Node.js. En primer lugar, se justificará el uso de Node.js y todos los módulos usados para la implementación, posteriormente se detallará las decisiones tomadas en cuanto a la configuración inicial del sistema se refiere, entonces, se explicará cómo se han generado las claves RSA y finalmente, se detallará cómo se han implementado cada uno de los roles que intervienen en el proceso.

4.1 Node.js

Node.js es un entorno de ejecución JavaScript de código abierto y multiplataforma que se utiliza para desarrollar aplicaciones escalables del lado del servidor y de red. Está basado en el motor de ejecución JavaScript V8 de Google Chrome.

El procesamiento de una solicitud con Node.js es eficiente y ligero. El software es adecuado para aplicaciones de uso intensivo de datos y en tiempo real, como chats en tiempo real, streaming de datos, proxies del lado del servidor, tableros de control del sistema, APIs REST y SPAs.

Por este motivo, es una de las mejores opciones para el desarrollo del esquema puesto que en este, se establecen varias conexiones cliente servidor, es frecuente el uso de comunicaciones y se trabaja con datos en tiempo real. [24] [25]

Además, se importarán los siguientes módulos:

Módulo net [26]**.** El módulo node:net proporciona una API de red asíncrona para crear servidores TCP o IPC basados en secuencias (net.createServer()) y clientes (net.createConnection()).

Módulo crypto [27]**.** El módulo node:crypto proporciona una funcionalidad criptográfica que incluye un conjunto de contenedores para las funciones hash, HMAC, cifrado, descifrado, firma y verificación de OpenSSL.

Módulo fs (File System) [28]**.** El módulo node:fs permite interactuar con el sistema de archivos proporcionando métodos para la lectura, escritura y creación de ficheros.

Módulo big-integer [29]**.** Es una biblioteca de enteros de longitud arbitraria para Javascript, que permite operaciones aritméticas en enteros de tamaño ilimitado, a pesar de las limitaciones de memoria y tiempo. Esta requiere ser instalada mediante npm puesto que no está integrada en Node.js.

Módulo os (operating system) [30]**.** El módulo node:os proporciona propiedades y métodos de utilidad relacionados con el sistema operativo.

4.2 Configuración del sistema

Los puntos a tener en cuenta antes de empezar a implementar el sistema son, generar y compartir las claves RSA que se usarán en el sistema, definir la función hash que se utilizará para generar los resúmenes de los conjuntos de puntos, así como, establecer el valor de los parámetros que conforman el esquema.

En cuanto a las claves RSA, existirá un pequeño script en Node.js que generará las claves y las volcará en un fichero para que los diferentes roles las usen. Nótese que los electores y los interventores solo deben tener acceso al fichero que contiene la clave pública puesto que, si obtuviesen acceso a la clave privada, el sistema de firmas ciegas se rompería y el esquema dejaría de ser seguro.

La función hash a utilizar en esta implementación será sha256, un algoritmo de generación de resúmenes bastante estándar desarrollado por la NSA que transforma una entrada dada en una salida de 256 bits. El uso de sha256 no es obligatorio, en otras implementaciones podría usarse otra función hash y esto no afectaría al sistema.

Sin pérdida de generalidad, asumiremos que los usuarios siempre generan un punto por interventor, así pues, en esta implementación, los usuarios generarían un número de puntos igual al número de interventores y compartirían cada punto que forma el conjunto con un interventor distinto. Los parámetros que formarían el esquema serían entonces un número de puntos p igual al número de interventores y un grado de la función polinómica $d = p - 1$. Obsérvese que, si no se asume que el número de puntos a generar es igual al del número de interventores, también se debería definir el parámetro de puntos máximos a generar para no romper con la imposibilidad de interpolar la función.

4.3 Claves RSA

Para generar las claves RSA se crea un script (RSA_Key_Generator.js). En este, se importará el módulo criptográfico crypto y se usará el método generateKeyPairSync(), pasándole como primer argumento ‘rsa’ para indicar el algoritmo usado para generar las claves y como segundo argumento el formato que poseerán estas claves. Para esta implementación se ha utilizado un tamaño de clave de 1024 bits para facilitar la lectura de los logs de salida de los diferentes roles, aunque el estándar es de 2048. Además de esto, se ha decidido usar el formato de claves PEM (correo de privacidad mejorada). El formato PEM se utiliza a menudo para representar certificados, solicitudes de certificados, cadenas de certificados y claves. Una vez las claves han sido generadas con el formato correcto se guardan en dos archivos: private_key.pem, al cual tendrá acceso solo la autoridad de identificación y public_key.pem, al cual tendrán acceso todos los roles que participan en la votación.

Ejemplo del contenido de los ficheros generados en formato PEM:

public_Key.pem

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDG9foVjWp4fRXcVR/3LqWJSwNC

qNviMrpy/C5mbWe/Aki7wyC6Az+abQK8vSXYV2LNREcAFInwq/k8UWEga5ltk+Lp

jZrW6EeoyKwvvFESSD50hGWMkHkywVIi+LLnYgDU2LsQHMWr++mYcrltZ/1nLP+G

1vI+CYKAexj1cnSS2wIDAQAB

-----END PUBLIC KEY-----

private_Key.pem

-----BEGIN PRIVATE KEY-----

MIICeAIBADANBgkqhkiG9w0BAQEFAASCAmIwggJeAgEAAoGBAOEhFMMcdU37DkLb

6kQv58MJ3ydkB7MWuQHG1vGdwjtIDxnhaFa4ErMBUzFEruadKkmtpWzmD22ahX+u

iumfI03mX6SN7Jofwtrt0+zvceFDJy3wrr5ScHOMnTDFMcPGi9qULhRGPgzeHV2e

qeA6CzDt4rIWoapvhCeA/BWmna79AgMBAAECgYB5RkDQ+juWzm9YRO9l3AiVYHP9

3DzplIwFEQApGDrRhHfeETcLwAoSsNwH5l/xiyiEmC5NA4G8IUlZBsV+JGP3SESH

aaZ/X4RxrgS0r19gMJsvWlkdlra1/zsAEKCZstMSAhklbQoPm2PLQ9IZIi/4SAC/

xYrORQSm9H62psjhYQJBAPCRW8QnfWTELFMwYhIUgPLJDg3d7mB0qP4ogMU2FS8R

hpDK1yXFDtmriOCEILUovVW57jktWDvAoeadzi0WvGcCQQDvkjAWw9QdpJVLLR+h

wjOWiSsobL6ozR3f1lL7oKWTRDo4GKQM7UT7i12IvS1q8ieK+8AqLviTzs4bkwu9

+5r7AkEAgEIvd9RKEXhZ3Xx1PQh7e3XgLVp+piRsmRoFEp/oCWVBpl7O+E8ozLqc

vn8TZSKoVolLDmbznN/mMas+6mEtzwJBAO5akCXWKFBl03O8LWCPCyyWxK2vo/Hm

QfbWEjHwHoxp8VKmkXGbD6sFecX2Eh35V4Zb+1D88qx8cWvV3TTkA8kCQQCBir09

6YkcrwwtWMGEWD1TDdzbc5zqn/WY+n2mBJDv+gp4PNcdJIP6PV5uRK746v6HEdNV

PzErZnEuybsB15SY

-----END PRIVATE KEY-----

4.4 Elector

Para el rol del elector, se creará un script llamado *User.js* que cumplirá las funciones descritas en el esquema para un votante. En primer lugar, se importarán los módulos necesarios, que, en este caso, son todos excepto el módulo *os*.

Una vez importados los módulos, se procederá a leer la clave pública del fichero *public_key.pem* mediante el módulo *fs*. A esta clave, hay que hacerle una conversión de formato *pem* a *jwk* (Un *json* que contiene los exponentes y el módulo) [31] haciendo uso de la librería *crypto*, para poder extraer el valor del módulo *n* y del exponente público *e*. Estos dos valores, son los parámetros que se van a utilizar en las ecuaciones de aritmética modular descritas previamente para el elector, y, deberán pasarse de base64 a hexadecimal para posteriormente convertirse en enteros grandes usando el módulo *big-integer*.

En cuanto a la máscara, se generarán números aleatorios de 1 a *n* hasta que uno sea invertible. Para que esto suceda, el máximo común divisor del número generado y del módulo *n* debe ser igual a 1.

En esta implementación, se considerará que la identidad y el voto son dos parámetros que se recibirán por consola. La identidad será una cadena igual al argumento recibido, mientras que, el voto se procesará para convertirlo de una cadena en base64 a una cadena en hexadecimal y posteriormente a decimal para poder concebir el voto como el término independiente de una función polinómica.

Una vez ya se posean todos estos valores iniciales, lo primero que deberá realizar el script del elector es leer del fichero *IpContainer.txt* las IPs de los interventores puesto que estos las habrán escrito en este fichero previamente. Estas IPs se guardarán en la variable global *listIP*.

Después de esto, el script esperará un tiempo de dos segundos y medio para evitar que, al leer el fichero de IPs, los partidos aún no las hayan escrito en el documento. Al esperarse el tiempo indicado, el script generará un número igual al número de IP en la lista de IPs menos uno de valores aleatorios entre 0 y 10000 (el rango puede ser el que se desee mientras que no sea superior al módulo *n*) mediante el método *randomBetween()* del módulo *big-integer* y los guardará en un array llamado coeficientes, estos valores son los que conformarán los coeficientes de la función polinómica.

Tras haber generado los coeficientes, se originará un punto por partido para construir el conjunto de puntos *P* guardándolos en un diccionario. Para la creación de cada punto, se generará un valor aleatorio entre 0 y 10000 (el rango puede ser el que se desee mientras que no sea superior al módulo *n*) que será la coordenada *x* del punto. Una vez la coordenada *x* esté generada, se le pasará como argumento a la función *poly(x)*. Esta función ha sido definida en el script y su objetivo es devolver el valor de salida del polinomio *q()* dada una *x*. Así pues, se usarán los coeficientes generados anteriormente y se procederá a calcular $y = q(x)$. Cuando ya tengamos el punto (x, y) se guardará como una entrada en el diccionario donde x es la clave e y es el valor. Este

proceso se repetirá un número de veces igual al número de IPs en la lista de IPs para que se genere un punto por partido.

Cabe recalcar que en Node.js los diccionarios se ordenan automáticamente por orden alfabético, por este motivo solo estamos generando puntos positivos puesto que los negativos concebidos como una cadena de texto no se ordenan automáticamente (debería hacerse manual).

Hecho esto, ya hemos construido la papeleta como un conjunto de puntos. Lo que se debe realizar ahora es aplicarle la función hash *sha256* al diccionario de puntos para construir el certificado sin enmascarar y sin firmar. Como hemos mencionado anteriormente, el diccionario se ordena automáticamente por lo que ya tendríamos la salida del *shash* que se detallaba en el algoritmo. Además, el valor del *shash* será guardado en hexadecimal puesto que esta es la forma más común.

Obtenido ya el valor del *shash*, se procederá a construir el certificado enmascarado b como se indica en el algoritmo. Para ello aplicaremos la fórmula:

$$b = shash(P) \cdot mask^{v} \; mod \; n$$

Para aplicar esta fórmula, se convertirá el *shash* de hexadecimal a decimal mediante enteros grandes. Posteriormente, se aplicará el método de exponenciación modular *modPow()* a la máscara elevándola a e y todo este cálculo, módulo n.

En este punto, el elector ya habría realizado todas las tareas previas a la comunicación con la autoridad, por lo que se procedería a enviar el compromiso enmascarado a esta para que lo certificara. Para enviar el compromiso enmascarado, se crea una conexión TCP mediante el método *createConnection()* del módulo *net*, pasándole como argumentos la IP de la autoridad y el puerto de conexión (En este caso 2000). La IP de la autoridad se guardará en una variable de entorno del sistema mediante Docker (detallado en la sección de Despliegue). Una vez establecida la conexión, se enviará la identidad del votante y el certificado sin firmar enmascarado b con el formato "*identidad;b*" y se esperará a recibir la respuesta de la autoridad.

El usuario puede recibir dos posibles valores de la autoridad de identificación. Uno de los dos posibles valores obtenidos es "Error". Esta recepción se da cuando el elector no está en la lista de votantes porque no está censado o porque ya ha votado. En este caso, el compromiso firmado es igual a "Error" y se cerrará la conexión con la autoridad para empezar a comunicarse con los partidos mediante la función *enviarPuntos (shash, bs)*. En la otra posible recepción, el usuario obtiene el compromiso firmado enmascarado en hexadecimal. Dada esta situación, el elector debe proceder con la eliminación de la máscara para guardar el compromiso firmado y enviarlo posteriormente a los partidos. Para eliminar la máscara se aplica la fórmula vista anteriormente:

$$b^{s} \cdot mask^{-1} \equiv shash(P)^{s} \cdot mask \cdot mask^{-1} \equiv shash(P)^{s} \; (mod \; n)$$

Con el objetivo de realizar esto, se convierte el dato recibido por la autoridad de hexadecimal a entero grande y se multiplica por el inverso modular de la máscara, que se calcula mediante el método *modInv()*, módulo *n*. Una vez hecho el cálculo, se vuelve a convertir el resultado de entero grande a hexadecimal para que el resultado se muestre como estándar y sea más legible. Hecho esto, se cierra la comunicación con la autoridad y se procede a comunicarse con los partidos llamando a la función *enviarPuntos(shash, bs)*.

En la función *enviarPuntos(shash, bs)*, se recorren todos los puntos en el diccionario de puntos mediante un bucle que recorre cada IP en la lista de IP's. En este bucle, se guarda la cordenada x y la cordenada y del punto, se codifica el mensaje que se va a enviar al partido con la IP de la iteración actual como: "*cordx;cordy;shash;compromiso_firmado*" y se procede a llamar al método *connectWithPartie()* pasando como argumentos el mensaje y la IP del partido al que se le envía el mensaje. Este proceso se realiza para cada partido porque el bucle recorre todas las IP de los partidos.

Por último, en el método *connectWithPartie()*, el elector crea una conexión cliente TCP con el partido mediante *createConnection()* pasándole como argumento la IP recibida como parámetro del método como hemos mencionado antes y el puerto de conexión (En este caso 2000). Una vez el partido haya recibido el punto, responderá al elector "recibido" y este cerrará la conexión con dicho partido. Cuando se haya cerrado la conexión con todos los partidos, el elector habrá finalizado con todas sus tareas, habrá participado en la votación y finalizará la ejecución del script.

4.5 Autoridad de identificación

El script *Authority.js* es el que realiza las tareas asignadas a la autoridad de identificación. Lo primero que se realiza en este script es importar los módulos necesarios, que, en este caso, son todos excepto el módulo *os*.

Después de esto, se procederá a leer la clave privada RSA mediante el módulo fs. A esta clave, hay que hacerle una conversión de formato *pem* a *jwk*, al igual que en el script del elector, pero en este caso se recupera el módulo y el exponente privado en lugar del público. Estos dos valores, son los parámetros que se van a utilizar en las ecuaciones de aritmética modular descritas previamente para la autoridad, y, deberán pasarse de base64 a hexadecimal para posteriormente convertirse en enteros grandes usando el módulo *big-integer*.

El script de la autoridad tendrá como variables globales: la lista de votantes censados, la hora de cierre de la votación y los minutos de cierre de la votación. La hora y los minutos (hh:mm) se recibirá como argumentos por la consola y se convertirán de cadena de texto a número.

Una vez establecidos todos los valores, la autoridad iniciará un servidor TCP que estará a la escucha en el puerto 2000. Este servidor, cuando reciba datos de un elector, deberá

separar el mensaje dividido por el separador punto y coma, en el que, la primera parte del mensaje es la identidad del votante y la segunda el compromiso enmascarado sin firmar. Tras obtener los datos, la autoridad comprobará que la identificación del elector está en la lista de votantes. Si el votante no se encuentra en la lista, la respuesta al elector será "Error" y se acabará la interacción. Por otra parte, si el votante se encuentra en la lista, entonces este se eliminará del registro para que no pueda volver a votar y se procederá a la firma del voto para convertirlo en el compromiso firmado enmascarado siguiendo la ecuación correspondiente.

$$b^s \equiv (shash(P) * mask^v)^s \, mod \, n$$

Para la firma, se convertirá el compromiso de hexadecimal a entero grande y se elevará a d mediante exponenciación modular (*modPow()*). El resultado de esta operación es un entero grande por lo que lo volveremos a convertir a hexadecimal. Cuando ya se haya obtenido el compromiso firmado enmascarado, se escribirá en el fichero *PBB_Autoridad.txt,* que actuará de boletín público, la identificación y el compromiso firmado enmascarado para que todo usuario pueda verificar su voto. Para acabar con la interacción con el elector, se le devolverá el compromiso firmado enmascarado.

El servidor se mantendrá a la escucha hasta que la hora de cierre llegue. Mientras esté activo, se realizará el proceso mencionado anteriormente para cada elector que establezca conexión y le envíe sus datos. Para cerrar el servidor, se comprueba en intervalos de 10 segundos mediante *setInterval()*, que la hora actual (h_ah_a:m_am_a) no sea igual a la hora que se le ha indicado por consola (hh:mm). Si se cumple esta condición, el servidor procederá a su cierre, se finalizará el script y los electores ya no podrán enviar sus compromisos para ser firmados.

4.6 Interventor

Para finalizar con la implementación, se detallará el script *Parties.js,* que cumplirá con la función del rol del interventor o partido. En este caso, todos los módulos mencionados con anterioridad son necesarios para la ejecución del programa.

Los interventores, son los únicos que hacen uso del módulo *os*. Es necesario que estos recuperen su IP para poder comunicarla a los electores y a los otros interventores. Por este motivo, se usa el módulo del sistema operativo para acceder a las diferentes interfaces de red y recuperar la información de la IP asociada al adaptador de red ethernet del dispositivo. En el caso de esta implementación, se consulta el adaptador de red Ethernet. Si los dispositivos estuviesen conectados por Wifi debería cambiarse la instrucción en la que se obtiene la IP.

El siguiente paso que siguen los interventores es el de recuperación de las claves RSA. El proceso es exactamente igual al del elector puesto que los interventores también necesitan el módulo y el exponente público para las ecuaciones modulares.

Para gestionar el cierre de las elecciones, el método usado es el mismo que en el caso de la autoridad. Se recibirán la hora y el minuto de finalización (hh:mm) mediante argumentos de consola y se ejecutará la comprobación mediante el *setInterval()*.

Los interventores proseguirán escribiendo sus IP en el fichero *IPContainer.txt*. Para ello, se hará uso del módulo *fs* y escribirán la IP recuperada como se explica anteriormente en el documento. Después de esto, los interventores esperarán dos segundos y medio para empezar a leer las IP's del documento, y a guardarlas en una variable global, puesto que estos deben comunicarse entre ellos más tarde. El tiempo de espera se debe a que todas las IP's deben estar escritas en el momento en el que se lean del fichero, y, si procediesen a leer el fichero nada más escribir la IP es posible que algún interventor aún no haya escrito la suya.

El interventor también utiliza dos sockets, uno para comunicarse con los usuarios, el cual se crea mediante el método *createServer()* del módulo net y se pone a la escucha en el puerto 2000, y, el otro para comunicarse con los otros interventores, que, se creará igual que el anterior pero esta vez se quedará a la escucha en el puerto 3000.

El servidor de comunicación Usuario-Partido está a la escucha para recibir mensajes de los usuarios que, como hemos mencionado en la sección del usuario, tienen el formato *"cordx;cordy;shash;compromiso_firmado"*. Como bien sabemos, el compromiso firmado puede ser igual a "Error" si la identidad no ha certificado su voto, por lo que, después de separar el mensaje recibido por cada punto y coma y guardarse *cordx*, *cordy*, *shash* y compromiso en cuatro diferentes variables, se pasará a comprobar que el compromiso es diferente a "*Error*". Si se cumple la condición, entonces los partidos pasan a verificar la firma elevando, mediante exponenciación modular, el compromiso firmado al exponente público de la clave RSA módulo n. Si el resultado es igual al shash entonces es correcta y se guardarán el punto en un diccionario en el que la clave es el compromiso y el valor un vector de puntos. El proceso de verificación es el que sigue:

$$shash(P) \equiv b^{sp} \ (mod\ n)$$

Si no se cumple la condición y el compromiso es igual a "*Error*", los interventores descartarán este punto. En cada uno de los casos el interventor responderá al usuario con "Recibido", "Firma incorrecta" o "Error votante no censado o duplicado" dependiendo de la situación.

Al ser Node.js asíncrono, los puntos se guardarán en dos diccionarios en lugar de uno. Cuando los partidos se comuniquen para compartir los puntos, es posible que algún partido haya recibido algún punto antes de enviar el suyo por lo que, al compartir el diccionario de puntos, estaría enviando el suyo y los ya recibidos. Para evitar esto, los puntos se guardarán en un diccionario que será el que se compartirá con los otros y en el otro que será en el que se guarden los diccionarios recibidos por los demás. Al final de este proceso se fusionarán los dos diccionarios.

Aclarado el punto anterior, cuando se cumple la hora de finalización establecida, los interventores cierran el servidor de comunicación con los usuarios y ya no serán aceptados más electores. Además, los interventores llamarán al método *partyToParty()* para empezar a compartir los puntos entre ellos.

Para compartir los puntos entre ellos, en el método partyToParty(), se crea una conexión cliente TCP para cada IP en la lista de IP's (excepto la IPpropia para no crear una conexión consigo mismo) en el puerto 3000 mediante *createConnection()* del módulo *net*. En cada una de estas conexiones, el interventor enviará el diccionario de puntos como un JSON y cuando reciba respuesta de confirmación cerrará esta conexión.

En cuanto a la parte de escucha, el interventor también recibirá los diccionarios de los demás interventores mediante el servidor de Partido-Partido que, como hemos dicho anteriormente, se encuentra a la escucha en el puerto 3000. Cada vez que este servidor recibe un mensaje (diccionario de otro interventor), el interventor convertirá los datos recibidos de un JSON a un diccionario y para cada clave en este diccionario, añadirá los puntos de los otros partidos a su diccionario auxiliar. El diccionario de puntos propio debería tener las mismas claves que el diccionario recibido, puesto que la clave es el certificado del voto que es igual para todos los puntos que conforman un mismo voto. Una vez hecho esto se responde al interventor con "Recibido" y se suma uno a la variable de recepciones. La variable de recepciones nos sirve para saber cuántos interventores nos han enviado ya sus diccionarios. Cuando esta variable sea igual al número de IP's en la lista menos uno (Para no contarse a sí mismo) entonces se cerrará el servidor de conexión entre partidos y se procederá al conteo de votos.

Al cerrarse la conexión, los partidos empezarán a reconstruir los puntos. Para esta reconstrucción se usará el método *interpolate(f,x)*. Este método está implementado en el propio script y se basa en la interpolación de Lagrange que hemos visto anteriormente:

$$q(x) = \sum_{i=0}^{j} y_i l_i(x),$$

donde:

$$l_i(x) = \prod_{\substack{0 \le k \le j \\ k \ne i}} \frac{x - x_k}{x_i - x_k}$$

Con el objetivo de implementar esta interpolación, se usará un bucle dentro de otro bucle para realizar el productorio dentro del sumatorio. Para ayudar a detallar cómo se ha realizado se mostrará una imagen con el código fuente:

```
//Interpolacion de lagrange para obtener el termino independiente
function interpolate(f,x)
{
    let result = 0;

    //Sumatorio(y*li(x))
    for (let i = 0; i < f.length; i++)
    {
        let term = f[i][1];
        //li(x) = Productorio((x-xk)/(xi-xk))
        for (let k = 0; k < f.length; k++)
        {
            if (k != i)
                term = term*(x - f[k][0]) / (f[i][0] - f[k][0])
        }
        result += term
    }

    return result;
}
```

Figura 4. Implementación de la interpolación en Node.js

Como podemos observar, a la función se le pasan dos argumentos. El primer argumento es f, que es una matriz de dos dimensiones en la que la primera dimensión se refiere a los puntos y la segunda dimensión a la coordenada del punto. El segundo argumento es x, que se refiere a la coordenada x del punto que se quiere reconstruir (Si se le pasa como argumento 0, entonces estaríamos recuperando el término independiente puesto que todos los coeficientes se volverían 0). Así pues, la función empieza por definir *resultado*, una variable que va guardando el resultado global. Para cada iteración, se define término, que, como en la fórmula empieza siendo igual a y_i equivalente a *f[i][1]* (coordenada 1 del punto i). Una vez ya se tiene el término, se entra en el bucle interno que cumple la función del productorio multiplicando el valor de *term* en la iteración anterior por *(x – f[k][0]) / (f[i][0] – f[k][0])* mientras que k ≠ i. Cuando el bucle interno termina, se suma a *result* el valor de *term* y se pasa a la siguiente iteración (Siguiente paso del sumatorio).

Una vez detallado cómo se ha implementado la interpolación, procederemos a explicar cómo recuperan los interventores los votos utilizando este método. En primer lugar, se recorren todos los conjuntos de puntos que hay en el diccionario, y, para cada conjunto de puntos se llama a la interpolación pasándole como argumento el vector de puntos y cero como coordenada x para así poder recuperar el coeficiente independiente. Antes de proseguir, se aplicará el método *Math.round()* a la salida de la función. Esto se debe a que, en los lenguajes de scripting estilo JavaScript, al hacer cálculos con enteros, puede que tengan cifras decimales minúsculas que en nuestro caso pueden molestar a la hora de convertir de decimal a hexadecimal para después recuperar el valor en base64.

Cabe aclarar que, al convertir el entero en una cadena hexadecimal, es posible que el primer cero a la izquierda desaparezca. Esto se debe a que los Buffers tienen pares de valores y añaden un cero a la izquierda si se queda un valor sin un par, mientras que, cuando se pasa de un número entero a hexadecimal los ceros a la izquierda no se tienen en cuenta. Para solucionar esto, tendremos que comprobar la longitud de la cadena hexadecimal. Si la cadena hexadecimal es par, significa que están todos los dígitos que la conforman, si es impar, deberemos añadir un cero a la izquierda para que el Buffer que reconstruye la cadena a base64 sea igual que el que la convirtió a hexadecimal y resulte en el mismo valor.

Una vez ya se tengan los valores en base64 recuperados, se escribirá en el boletín público de los partidos (*PBB_Partidos.txt*) el certificado firmado, el conjunto de puntos y el voto recuperado. Después de esto se comprueba si esa entrada existe en el diccionario de conteo de partidos, que, tiene como clave el nombre de los partidos y como valor la cantidad de votos sumados. Si la entrada ya existe se le suma uno a su valor, si no existe, se crea una nueva entrada con valor inicial 1.

Cuando termine la ejecución el resultado del diccionario será el resultado de la votación, representando como claves los partidos y como valor la cantidad de votos que ha recibido cada partido. Además, se habrá terminado el proceso completo de votación y escrutinio y se tendrán los boletines públicos para que todo el mundo pueda verificar que la votación se ha realizado correctamente.

5. Despliegue

En esta sección se abarcará el despliegue del esquema de votación. Para el despliegue, se ha decidido usar la herramienta Docker. Durante la sección, se detallará qué es Docker, por qué hemos escogido esta herramienta, cómo se ha realizado el despliegue de cada rol y, por último, el uso de DockerCompose para lanzar todos los servicios que conforman el protocolo de votación.

5.1 Docker

Docker es un sistema operativo (o runtime) para contenedores. El motor de Docker se instala en cada servidor en el que desee ejecutar contenedores y proporciona un conjunto sencillo de comandos que puede utilizar para crear, iniciar o detener contenedores. Además de esto posee muchas ventajas como: despliegue rápido, costo bajo, optimización de recursos y simplicidad de uso. La principal desventaja de Docker se presenta en despliegues muy grandes donde Kubernetes se muestra superior a este, pero, en el caso de nuestro proyecto Docker no sufre estos inconvenientes puesto que es un pequeño despliegue de prueba, y, además, la gestión de contenedores en Kubernetes es bastante más compleja y se necesita un orquestador.

En el caso de este proyecto, se ha decidido usar Docker por su facilidad y porque Docker puede lanzar más de un contenedor en la misma máquina, gestionando automáticamente la red que conecta estos contenedores y los contenedores en sí.

Además de esto, Docker posee la herramienta DockerCompose. Esta herramienta se desarrolló para ayudar a definir y compartir aplicaciones de varios contenedores. Con Compose, podemos crear un archivo YAML para definir los servicios y, con un solo comando, podemos lanzar y eliminar todos los contenedores que forman la aplicación a la vez sin necesidad de ir uno por uno. [32]

5.2 Dockerfile

Un Dockerfile es un archivo de texto plano que contiene una serie de instrucciones necesarias para crear una imagen que contendrá todo lo necesario para ejecutar la aplicación que se desee. La base del funcionamiento de Docker es mediante Dockerfiles.

En el contexto de este proyecto se van a realizar tres Dockerfiles diferentes: el del elector, el de la autoridad y el de los interventores. Las tres imágenes van a estar basadas en la última versión de la imagen node oficial de Docker. Esta es una imagen que tiene ya instalado y configurado todo lo necesario para ejecutar scripts en Node.js.

5.2.1 Elector

En el Dockerfile del elector se empezará por seleccionar la imagen base de Node.js mediante la instrucción “FROM node:latest”.

Una vez escogida la imagen base, se establece cual será el directorio de trabajo en el contenedor, es decir, donde el contenedor copiará y buscará todos los archivos por defecto. Este directorio será /uservoting y se establece a través de la instrucción “WORKDIR /uservoting”.

El siguiente paso es copiar los ficheros necesarios al directorio de trabajo, que, en el caso del elector es el script *User.js* y *public_key.pem*. Estos dos ficheros se copiarán mediante las instrucciones “COPY User.js ./” y “COPY public_key.pem ./”.

Después de esto, se instalarán las dependencias necesarias, que, para este proyecto es el módulo *big-integer*. Esto se realizará con la instrucción “RUN npm install big-integer”.

Por último, se establece el comando por defecto para iniciar la aplicación (node User.js) por medio de la instrucción “ENTRYPOINT ["node", "User.js"]”.

5.2.2 Autoridad

Al igual que en el caso del elector, la autoridad seleccionará la última versión de la imagen de node a través de la instrucción “FROM node:latest”.

Después de esto, se establece cual será el directorio de trabajo en el contenedor que será el mismo que en el caso del elector. Este directorio será /uservoting y se establece a través de la instrucción “WORKDIR /uservoting”.

En el Dockerfile de la autoridad se copiarán los ficheros *Authority.js*, *public_key.pem* y *private_key.pem* al directorio de trabajo. Estos tres ficheros se copiarán mediante las instrucciones “COPY Authority.js ./”, “COPY public_key.pem ./” y “COPY private_key.pem ./”.

En este caso también se instalará la dependencia del módulo *big-integer*. La instrucción “RUN npm install big-integer” es la que se encarga de esto.

Para finalizar, definiremos el comando por defecto para iniciar la aplicación. En este caso es node Authority.js y se fija con la instrucción “ENTRYPOINT ["node", "Authority.js"]”.

5.2.3 Interventor

El interventor también usa la última versión de la imagen de base de node por lo que también utilizará la instrucción “FROM node:latest”.

El directorio de trabajo del interventor será el mismo que hemos comentado para los otros dos roles. Este directorio era /uservoting y se conseguía mediante la instrucción “WORKDIR /uservoting”.

En el caso del interventor, los ficheros a copiar al directorio de trabajo son *Parties.js* y *public_key.pem*. Las instrucciones a introducir para la copia de estos archivos en el Dockerfile del interventor son “COPY Parties.js ./” y “COPY public_key.pem ./”.

En este Dockerfile también se instalará el módulo *big-integer* por medio de la instrucción “RUN npm install big-integer”.

Por último, el comando por defecto para iniciar el script del interventor es node Parties.js por lo que, al igual que en los Dockerfiles anteriores, se establecerá con la instrucción “ENTRYPOINT ["node", "Parties.js"]”.

5.3 DockerCompose

Como hemos dicho anteriormente, el *docker-compose.yml* es un fichero YAML en el que se definen todos los servicios que se van a levantar cuando se ejecute el comando docker compose up. Además, esta herramienta también permite la creación de volúmenes para compartir datos entre contenedores. En el caso de este despliegue, se van a definir tres servicios distintos de los cuales se lanzarán distintas instancias. Los tres tipos de servicios son: el servicio de tipo usuario, el servicio de tipo partido y el servicio de tipo autoridad.

Nótese que el servicio de usuario se debe definir para cada usuario que participe en la votación mientras que el servicio del interventor basta con definirlo una vez e indicar mediante un parámetro que se va a lanzar más de uno. Esto se debe a que los argumentos que recibe el script son distintos para cada elector. Sin embargo, no significa que sean servicios completamente distintos puesto que son todos iguales exceptuando los argumentos recibidos por consola.

En el fichero, lo primero que tenemos que indicar es la versión que se va a utilizar. En nuestro caso es la “3.9”. Una vez escogida la versión, se abre la sección de servicios para definir cada servicio que va a lanzar en el despliegue. En cada servicio se deben indicar los parámetros de configuración como la imagen sobre la que se construirá, de qué otros servicios depende, argumentos para el comando de lanzamiento, etc. Esto se realiza mediante diferentes subsecciones que hay en cada servicio. En lo que resta de sección se abarcarán los tres tipos de servicio que se van a utilizar en este proyecto y sus configuraciones.

5.3.1 Elector

El servicio del elector se construye en base a la imagen definida anteriormente en el Dockerfile del elector {5.2.1}. Para indicarle a Docker que la imagen en la que se basa este servicio es la mencionada anteriormente se debe indicar mediante build el fichero en el que se encuentra este Dockerfile.

Una vez indicada la imagen, se establecen las dependencias de este servicio. El servicio del elector depende del servicio de la autoridad y del de los interventores puesto que cuando el usuario empiece el proceso de construcción y envío de su voto la autoridad y los partidos ya deben estar en marcha y a la escucha de recibir las peticiones de los usuarios.

Anteriormente, en la sección de implementación, se ha mencionado que el elector conocía la IP de la autoridad porque la tenía guardada en las variables de entorno. En este punto es en el que se establece que, en las variables de entorno del contenedor que se va a desplegar, debe establecerse el valor de la variable de entorno IP_AUTORIDAD que será igual al nombre del servicio de la autoridad. De esta manera, sea cual sea la IP que se le asigne al servicio de la autoridad, los electores siempre la conocerán.

Otro aspecto a tener en cuenta es el de los volúmenes. El elector necesita tener acceso al fichero en el que se escriben las IPs para poder leerlas. Para que esto sea posible, se crea el volumen y se indica dónde está el fichero en la máquina host y donde se colocará en el árbol de directorios del contenedor.

Por último, se indicará mediante la opción command, los argumentos necesarios para la ejecución por consola del comando. En el caso del servicio del elector estos argumentos son el voto y la identidad.

5.3.2 Autoridad

En el parámetro build de la autoridad le indicaremos que la imagen base que se va a usar para construir este servicio es la del Dockerfile que hemos definido anteriormente para la autoridad {5.2.2}.

Para el caso de la autoridad no depende de ningún otro servicio. Este servicio puede lanzarse como primera instancia ya que no necesita nada de los otros servicios. Además, tampoco necesita definir ninguna variable de entorno al contrario que en el caso del elector.

Para los volúmenes, la autoridad necesitará tener acceso al fichero que se usará como boletín público que en este caso será *PBB_Autoridad.txt*. Así pues, se le indicará al servicio donde se encuentra este fichero en la máquina host de los contenedores y también donde debe montarlo dentro del contenedor.

Por último, los únicos argumentos que usa el servicio de la autoridad son la hora y los minutos de finalización.

5.3.2 Interventor

La imagen base de los partidos será la definida previamente para los interventores {5.2.3}. Al igual que en los otros servicios, se le indicará en qué lugar se encuentra el Dockerfile de los interventores.

Los interventores tampoco dependen de ningún otro servicio por lo que pueden ser lanzados en cualquier orden mientras se lancen antes de los que sí dependen de ellos. En el caso de los interventores, tampoco es necesaria ninguna variable de entorno.

En el caso de este servicio, los volúmenes necesarios que deberán ser montados en el contenedor son: el documento en el que se van a guardar y a leer las IPs *IPContainer.txt* y el boletín público en el que se publicarán los datos de los partidos para poder ser verificados. Para ello, se le indicará la ubicación en la máquina host y en donde se volcará en el contenedor.

Al igual que en el caso de la autoridad, los únicos argumentos necesarios para el funcionamiento de los partidos es la hora y los minutos de finalización del proceso de votación.

Aparte de todos estos parámetros, puede lanzarse más de un contenedor de los partidos mediante el parámetro scale. Al contrario que en los usuarios, esto puede realizarse porque todos los parámetros son iguales para los interventores. Esto nos resultará útil para lanzar todos los interventores que participan en la votación sin necesidad de definir cada uno.

6. Pruebas

Una vez detallada la implementación {4} y el despliegue {5}, procederemos a realizar pruebas para demostrar que, en efecto, todo está correctamente realizado y el esquema funciona de manera eficiente y precisa. Para ello, se presentará primero una simulación de votación para después mostrar los resultados de esta simulación y detallarlos.

6.1 Simulación de votación.

Para la realización de las pruebas, simularemos una situación real en la que existen cuatro usuarios, dos partidos y la autoridad de identificación.

En estas elecciones se presentan dos partidos que son el *"Amarillo"* y el *"Verde"*. Los usuarios "*11111111A*", "*22222222B*", "*33333333C*" y "*44444444D*" quieren votar en las elecciones, pero en la lista del censo solo están los usuarios "*11111111A*", "*22222222B*" y "*33333333C*".

El objetivo de introducir cuatro usuarios es testear cada uno de los casos posibles: votar al partido *"Amarillo"*, votar al partido *"Verde"*, no votar, intentar votar sin estar censado e intentar votar dos veces. Para probar estos casos, el usuario *"11111111A"* intentará votar dos veces al partido *"Amarillo"* (para ello introduciremos el valor dos en el parámetro scale de este servicio en el *"dockercompose.yml"* para que este servicio se lance dos veces), el usuario "*22222222B*" votará una sola vez al partido *"Verde"*, el usuario "*33333333C*" decide no votar en estas elecciones y el usuario "*44444444D*" quiere votar al partido "Amarillo".

Presentada ya esta situación, se construye el fichero YAML como se ha detallado anteriormente {5.3} y se lanza mediante docker compose up. Esto provoca la creación de siete contenedores. El contenedor de la autoridad, el contendor del partido *"Amarillo"*, el contenedor del partido *"Verde"*, dos contenedores del usuario *"11111111A"* porque va a intentar votar dos veces, el contenedor del usuario "*22222222B*" y el contenedor del usuario "*44444444D*".

Durante la implementación de los roles, se han ido introduciendo impresiones por consola para poder ver el comportamiento de estos al realizar las pruebas. Una vez finalizada la ejecución de todos los contenedores, podremos acceder a sus logs para ver las impresiones que estos han realizado y analizar el comportamiento de todos los elementos que han participado en esta votación.

Nótese que no se lanza el contenedor del usuario *"3333333333C"* puesto que, como hemos mencionado, este no va a participar en la votación. Por este motivo, cuando analicemos los logs, en el caso de este usuario no tendremos nada que analizar, pero podremos observar que su ausencia en la votación no afecta a esta misma.

6.2 Análisis de los resultados

Procedemos pues a analizar las diferentes salidas de cada elemento y a demostrar que la implementación funciona.

6.2.1 Usuario 11111111A

Como bien sabemos, en este usuario debemos analizar la salida de dos contenedores. En primer lugar, analizaremos su primer intento de voto para después proceder con el segundo intento. Vamos a ver la salida y a analizarla:

```
-MODULO:
15809111669664963347198832277008859023553174439045376884999594552195117882199912296227654151223003273473139498
24625903130068316275779072318813516610451754836205154451565629075049929397190613233438518474088717779927456127
03501600882663923141457968929560408150741172100231933679164575608410076076981511792733949
-EXPONENTE PUBLICO: 65537
-MASCARA:
15806905186116700906223526971356063216378324817769583256358621288460175190459763695741744988480548375089794929
67886397254687931302287064867472698282169282428010652245083962887854811304570527769210957058956585724824067905
94703748151910615922851030954301630027933432108008402318223373396031587066642160344245352

-IDENTIFICACION: 11111111A
-VOTO: Amarillo
-VOTO COMO ENTERO: 2639987890536

-LISTA DE IP: 192.168.16.3,192.168.16.4

-COEFICIENTES GENERADOS: 9582
-PUNTOS: {"3042":2640017038980,"9997":2640083681790}

-HASH DEL VOTO EN HEXADECIMAL: 4374ff9964febe2d95e6880d06f5e2d9a66d9269a5ebefcdcc1ab3e4e178b993
-VOTO ENMASCARADO EN HEXADECIMAL:
63f12b1a4983c26b5a9da5de0f2aab9b9d84e8b3847ec891e8f914891f1e1618d5292176b5142cd26189d84e77b8c737dde3531cc68926
f490ab80058bf4a8e537582c1b2833163d767616673b3b5f6a548b64b5413c9fd985b375d8f88190726b69a40723b6c18266938a684516
afbb54d512a64e7a4a64e6b307489133a5fd

-CONECTADO A LA AUTORIDAD
-VOTO FIRMADO CON MASCARA:
9a2c80198aea8ecebc431cd60894ec79015b59d40cbba4fcc0ee5677732090cf2e9996f16ce39eb0c22873f592d17d1eea1751a686e6f1
947d8b31af11bba16eeb90b8838088513ffff898150361cbdaa4bb461da58bd2dd14dba5364d21489ed626995aa3025b85335d8461862b
947953472b662fda097d6809502c7eb95d18
-VOTO FIRMADO SIN MASCARA:
cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e56
4cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef93718
94c099f2f963f7f82fce093868d5cd7cc7b1
-FIN DE CONEXION CON LA AUTORIDAD

-CONECTADO AL PARTIDO 192.168.16.3
-CONECTADO AL PARTIDO 192.168.16.4
-RECIBIDO
-FIN DE CONEXION CON EL PARTIDO
-RECIBIDO
-FIN DE CONEXION CON EL PARTIDO
```

Figura 5. Log del primer intento de votación de 11111111A

Podemos observar los parámetros RSA, la información del votante y cómo se han generado los puntos. Vemos pues que, la primera vez que intenta votar, todo está correcto puesto que recibe la certificación de la autoridad y los partidos aceptan los puntos que les envía. Vamos a ver qué ocurre cuando intenta votar por segunda vez.

```
-MODULO:
158091116696649633471988322770088590235531744390453768849995945521951178821999122962276541512230032734731394982462590313006831627577907231881351661045175483620515445156562907504992939719061323343851847408871777992745612703501600882663923141457968929560408150741172100231933679164575608410076076981511792733949
-EXPONENTE PUBLICO: 65537
-MASCARA:
104065654986569511593526807761497365080421330729659141025046334587045338740847065570372406524924814610318442144476400188337208437132737756310372258080480568382501211392532131510401127921462910833579804685141898141652645308339592154673773944484962589452034500227277904456687194991793520192836642506139094614449

-IDENTIFICACION: 11111111A
-VOTO: Amarillo
-VOTO COMO ENTERO: 2639987890536

-LISTA DE IP: 192.168.16.3,192.168.16.4

-COEFICIENTES GENERADOS: 777
-PUNTOS: {"2337":2639989706385,"6263":2639992756887}

-HASH DEL VOTO EN HEXADECIMAL: 1a6b076cc8a4eb5803d8cdcdaba00189ab8c20837b0fa961b2a30bd916795b73
-VOTO ENMASCARADO EN HEXADECIMAL:
b87f02e06bbea657c2208429fdc9501926e1d1188175e4ac97c0fc89142347febdb54bf49d975f97a9446f8c24a450b24d7bd30f2dcacb863cfd9917371b556b767f3adada694d2f9dd1d3f2d756253bf1906b416caba1691014255633e592199586156cb15e59594556b8a01492fd86d4f271b99baa98623684c84dfed31355

-CONECTADO A LA AUTORIDAD
-FIRMA: Error
-FIN DE CONEXION CON LA AUTORIDAD

-CONECTADO AL PARTIDO 192.168.16.3
-CONECTADO AL PARTIDO 192.168.16.4
-ERROR VOTANTE NO CENSADO O DUPLICADO
-ERROR VOTANTE NO CENSADO O DUPLICADO
-FIN DE CONEXION CON EL PARTIDO
-FIN DE CONEXION CON EL PARTIDO
```

Figura 6. Log del segundo intento de votación de 11111111A

Esta vez vemos como es el mismo usuario, pero la respuesta de la autoridad ahora ha sido Error. Además, los partidos la han respondido con el mensaje de votante no censado o duplicado.

En este caso, el esquema está funcionando como se esperaba. Ha aceptado su primer voto porque está en el censo, pero, cuando ha intentado votar por segunda vez no ha aceptado su voto.

Obsérvese que, aunque sea el mismo usuario, la generación de coeficientes y puntos es totalmente diferente puesto que el intento de votar dos veces no se realiza mediante intentar enviar dos veces el voto una vez construido, sino que se intentan construir dos papeletas diferentes para votar dos veces. Si el elector intentase enviar el voto con el mismo certificado para que este se contase dos veces esto no funcionaría puesto que en el diccionario de puntos de los partidos tendría la misma clave y lo único que provocaría es que en lugar de tener un punto por partido ese voto tendría dos puntos por partido.

6.2.2 Usuario 22222222B

El usuario *"22222222B"* va a realizar el proceso de votación normal y va a votar al partido *"Verde"*. Procedemos a observar la salida de este caso y a analizar el resultado obtenido.

```
-MODULO:
158091116696649633471988322770088590235531744390453768849995945521951178821999122962276541512230032734731394982462590313006831627577907231881351661045175483620515445156562907504992939719061323343851847408871777992745612703501600882663923141457968929560408150741172100231933679164575608410076076981511792733949
-EXPONENTE PUBLICO: 65537
-MASCARA:
665130033188346444921016428148584434859453686712395295667324865492882787642072762877535718955065896005087575321753438651414907020072329746208174944327122635723067634395741210127668609042501928093353423121023892425808694339663206784216275618422132382134728638899810974723567489512570837271973510241981441933555

-IDENTIFICACION: 22222222B
-VOTO: Verde
-VOTO COMO ENTERO: 5630685

-LISTA DE IP: 192.168.16.3,192.168.16.4

-COEFICIENTES GENERADOS: 9624
-PUNTOS: {"1059":15822501,"9488":96943197}

-HASH DEL VOTO EN HEXADECIMAL: aed50e19b2a31762559e0b04557fe2695ad3217a6063307497d1a34664530b3f
-VOTO ENMASCARADO EN HEXADECIMAL:
51ebf0c4a9a18cb48eb352a4b04add34f9d528a8abd42430d953e43a5f84b42e33b4e7f2239afd416a567333b241f2ff5ddaf9d358ebdc0c2118ad4794df0f8b8a754859b1975b27cfe26638ec02adcda8ea05499459cf94d7327a4af9d76aff23c95afff6ba2534cab655a4d35ae92f552631c100b3623ebfd902e901a37156

-CONECTADO A LA AUTORIDAD
-VOTO FIRMADO CON MASCARA:
42e3ee45b2bffe6cf0be969338889b497faed729369ce351b04082a71f6441ed7b7e95444c163b5d78f44f85d1fb7461e3fd7a3e496d54a482c0abac3636af6f935cc23ff76d6557155b6cabf56bb90d56641c0dddc6e36b11cbfee07393ade57db11c41f87c56a19d0a7a8a8990ede4b08c729c0e555e866955ba5e0c80f7f8
-VOTO FIRMADO SIN MASCARA:
2f92bfab6e8f5a26f4914ba7aad3d90176786fa8e62d2b11acc5b215a972eca3acfd5f9ddd570a67807ba8f139639102888d500cffc834317c6580ca46c2d63efbdce640cb40b6061305a09fd54a695b39be9322b6f185bd8729fc1697c97fdd74511a0fc82cd0fb8a07bcc957d46a3d17a9d9778d301c6df996a38ef76ec410
-FIN DE CONEXION CON LA AUTORIDAD

-CONECTADO AL PARTIDO 192.168.16.3
-CONECTADO AL PARTIDO 192.168.16.4
-RECIBIDO
-FIN DE CONEXION CON EL PARTIDO
-RECIBIDO
-FIN DE CONEXION CON EL PARTIDO
```

Figura 7. Log votación de 22222222B

Vemos que en este caso el usuario ha votado al partido *"Verde"*. Los parámetros RSA y la lista de IP es igual para todos los usuarios, pero en este caso cambia el voto, la generación de los puntos y los certificados. El elector *"22222222B"* también ha podido obtener su certificado y los partidos han aceptado sus puntos.

El resultado también ha sido el esperado para este elector. Ha votado solamente una vez y como estaba en el censo de votantes su voto ha sido aceptado.

Nótese como cambia el diccionario de puntos y como, si se calcula, este siempre coincide con el voto en forma decimal más la coordenada x del punto por su correspondiente coeficiente (En esta situación solo se genera un coeficiente porque solo hay dos partidos).

6.2.3 Usuario 44444444D

Este usuario no está censado por lo que veremos cómo responde el sistema cuando un usuario no censado intenta votar. Acto seguido, veremos la salida para el caso de *"44444444D"* y procederemos a estudiarla.

```
-MODULO:
158091116696649633471988322770088590235531744390453768849995945521951178821999122962276541512230032734731394982462590313006831627577907231881351661045175483620515445156562907504992939719061323343851847408871777992745612703501600882663923141457968929560408150741172100231933679164575608410076076981511792733949
-EXPONENTE PUBLICO: 65537
-MASCARA:
653072967546443720034217012226304643641718740537661914670216328849833401029959429465779837598198162822971418821518369065625730918026698768598542096350244111328746619986893895095411824108518175288404060479643603181180083094849065277321686205612046882582348936926434535333441420468396118887363331240125345556 9

-IDENTIFICACION: 44444444D
-VOTO: Amarillo
-VOTO COMO ENTERO: 2639987890536

-LISTA DE IP: 192.168.16.3,192.168.16.4

-COEFICIENTES GENERADOS: 9709
-PUNTOS: {"2912":2640016163144,"3084":2640017833092}

-HASH DEL VOTO EN HEXADECIMAL: 297e8dc50f6a3cb898b044e34e28b5b35b6b83ef388daabe9c7f0ca63b8d25cf
-VOTO ENMASCARADO EN HEXADECIMAL:
b006c6b08a2affd8dbd2a2af2f1508e5ecd25d4b1c50865856f77494b3f7bdc1827c791602da4eb6389bd5f8c364755fd978e3bde62f72582fcf475487c1b32d4d625affdbdb697c7b34fb2a264905ce325401dfe28451d1b9300088ef6a4dbc4fac940516d3306859362fd6958ba2b7b36bced17524d5d2b9b24ecc57917597

-CONECTADO A LA AUTORIDAD
-FIRMA: Error
-FIN DE CONEXION CON LA AUTORIDAD

-CONECTADO AL PARTIDO 192.168.16.3
-CONECTADO AL PARTIDO 192.168.16.4
-ERROR VOTANTE NO CENSADO O DUPLICADO
-ERROR VOTANTE NO CENSADO O DUPLICADO
-FIN DE CONEXION CON EL PARTIDO
-FIN DE CONEXION CON EL PARTIDO
```

Figura 8. Log intento de votación de elector no censado.

La respuesta por parte de la autoridad y de los partidos ha sido exactamente la misma que en el caso del usuario que ha intentado votar dos veces en su segundo intento. Concluimos pues que para este caso el esquema también responde como es debido.

6.2.4 Autoridad

Llegados a este punto, vamos a ver como gestiona la autoridad de identificación todas las peticiones que le llegan para certificar las papeletas.

```
-MODULO:
158091116696649633471988322770088590235531744390453768849995945521951178821999122962276541512230032734731394982462590313006831627577907231881351661045175483620515445156562907504992939719061323343851847408871777992745612703501600882663923141457968929560408150741172100231933679164575608410076076981511792733949
-EXPONENTE PRIVADO:
85161798432313329265835722524302418323621964752133601713234613435265032228082717198837466187768269461021364549199067525373349158185611744729530479599903480262326467992300386889070751020942376255782932229272914466372704722296438145496123535856698804193220294909653878816377829677486212145284774671357730939233

-LISTA DE VOTANTES: 11111111A,22222222B,33333333C

-SERVIDOR TCP INICIADO EN EL PUERTO 2000

-USUARIO 44444444D CONECTADO
-DATOS RECIBIDOS DEL USUARIO:
b006c6b08a2affd8dbd2a2af2f1508e5ecd25d4b1c50865856f77494b3f7bdc1827c791602da4eb6389bd5f8c364755fd978e3bde62f72582fcf475487c1b32d4d625affdbdb697c7b34fb2a264905ce325401dfe28451d1b9300088ef6a4dbc4fac940516d3306859362fd6958ba2b7b36bced17524d5d2b9b24ecc57917597
-USUARIO 44444444D NO ESTA EN LA LISTA DEL CENSO
-USUARIO 44444444D DESCONECTADO

-USUARIO 11111111A CONECTADO
-DATOS RECIBIDOS DEL USUARIO:
63f12b1a4983c26b5a9da5de0f2aab9b9d84e8b3847ec891e8f914891f1e1618d5292176b5142cd26189d84e77b8c737dde3531cc68926f490ab80058bf4a8e537582c1b2833163d767616673b3b5f6a548b64b5413c9fd985b375d8f88190726b69a40723b6c18266938a684516afbb54d512a64e7a4a64e6b307489133a5fd
-VOTO FIRMADO POR LA AUTORIDAD:
9a2c80198aea8ecebc431cd60894ec79015b59d40cbba4fcc0ee5677732090cf2e9996f16ce39eb0c22873f592d17d1eea1751a686e6f1947d8b31af11bba16eeb90b8838088513ffff898150361cbdaa4bb461da58bd2dd14dba5364d21489ed626995aa3025b85335d8461862b947953472b662fda097d6809502c7eb95d18
-USUARIO 11111111A DESCONECTADO

-USUARIO 22222222B CONECTADO
-DATOS RECIBIDOS DEL USUARIO:
51ebf0c4a9a18cb48eb352a4b04add34f9d528a8abd42430d953e43a5f84b42e33b4e7f2239afd416a567333b241f2ff5ddaf9d358ebdc0c2118ad4794df0f8b8a754859b1975b27cfe26638ec02adcda8ea05499459cf94d7327a4af9d76aff23c95afff6ba2534cab655a4d35ae92f552631c100b3623ebfd902e901a37156
-VOTO FIRMADO POR LA AUTORIDAD:
42e3ee45b2bffe6cf0be969338889b497faed729369ce351b04082a71f6441ed7b7e95444c163b5d78f44f85d1fb7461e3fd7a3e496d54a482c0abac3636af6f935cc23ff76d6557155b6cabf56bb90d56641c0dddc6e36b11cbfee07393ade57db11c41f87c56a19d0a7a8a8990ede4b08c729c0e555e866955ba5e0c80f7f8
-USUARIO 22222222B DESCONECTADO

-USUARIO 11111111A CONECTADO
-DATOS RECIBIDOS DEL USUARIO:
b87f02e06bbea657c2208429fdc9501926e1d1188175e4ac97c0fc89142347febdb54bf49d975f97a9446f8c24a450b24d7bd30f2dcacb863cfd9917371b556b767f3adada694d2f9dd1d3f2d756253bf1906b416caba1691014255633e592199586156cb15e59594556b8a01492fd86d4f271b99baa98623684c84dfed31355
-USUARIO 11111111A NO ESTA EN LA LISTA DEL CENSO
-USUARIO 11111111A DESCONECTADO

-SERVIDOR CERRADO
```

Figura 9. Log de la autoridad de identificación.

En cuanto a los parámetros se refiere, podemos observar que el módulo es igual que en el caso de los usuarios, pero esta vez se muestra el exponente privado (recuérdese que estas trazas se están ofreciendo con fines ilustrativos y jamás se generarían en un sistema en producción) en lugar del público. También podemos observar la lista del censo de votantes.

En lo que se refiere a las peticiones, podemos observar cómo se conecta cada usuario, los datos que recibe la autoridad, el certificado firmado y cómo cada usuario se desconecta tras haber recibido su certificado.

Observamos como la autoridad cumple con su función a la perfección, permitiendo el voto al elector *"11111111A"* y *"22222222B"*, denegando el voto al elector *"44444444D"*, y, no permitiendo votar por segunda vez al elector *"11111111A"*.

6.2.5 Partido Amarillo

Pasamos pues a ver y detallar los registros de la consola del partido *"Amarillo"*.

```
-IP PROPIA: 192.168.16.4

-MODULO:
158091116696649633471988322770088590235531744390453768849995945521951178821999122962276541512230032734731394982462590313006831627577907231881351661045175483620515445156562907504992939719061323343851847408871777992745612703501600882663923141457968929560408150741172100231933679164575608410076076981511792733949
-EXPONENTE PUBLICO: 65537

-SERVIDOR TCP INICIADO EN EL PUERTO 2000
-SERVIDOR TCP INICIADO EN EL PUERTO 3000

-LISTA DE IP: 192.168.16.3,192.168.16.4

-USUARIO CONECTADO
-DATOS RECIBIDOS: 3084;2640017833092;297e8dc50f6a3cb898b044e34e28b5b35b6b83ef388daabe9c7f0ca63b8d25cf;Error
-USUARIO DESCONECTADO

-USUARIO CONECTADO
-DATOS RECIBIDOS:
9997;2640083681790;4374ff9964febe2d95e6880d06f5e2d9a66d9269a5ebefcdcc1ab3e4e178b993;cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1
-VERIFICACION: true
-PUNTOS:
{"cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1":[[9997,2640083681790]]}
-USUARIO DESCONECTADO

-USUARIO CONECTADO
-DATOS RECIBIDOS:
9488;96943197;aed50e19b2a31762559e0b04557fe2695ad3217a6063307497d1a34664530b3f;2f92bfab6e8f5a26f4914ba7aad3d90176786fa8e62d2b11acc5b215a972eca3acfd5f9ddd570a67807ba8f139639102888d500cffc834317c6580ca46c2d63efbdce640cb40b6061305a09fd54a695b39be9322b6f185bd8729fc1697c97fdd74511a0fc82cd0fb8a07bcc957d46a3d17a9d9778d301c6df996a38ef76ec410
-VERIFICACION: true
-PUNTOS:
{"cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1":
[[9997,2640083681790]],"2f92bfab6e8f5a26f4914ba7aad3d90176786fa8e62d2b11acc5b215a972eca3acfd5f9ddd570a67807ba8f139639102888d500cffc834317c6580ca46c2d63efbdce640cb40b6061305a09fd54a695b39be9322b6f185bd8729fc1697c97fdd745111a0fc82cd0fb8a07bcc957d46a3d17a9d9778d301c6df996a38ef76ec410":[[9488,96943197]]}
-USUARIO DESCONECTADO

-USUARIO CONECTADO
-DATOS RECIBIDOS: 6263;2639992756887;1a6b076cc8a4eb5803d8cdcdaba00189ab8c20837b0fa961b2a30bd916795b73;Error
-USUARIO DESCONECTADO

-DATOS RECIBIDOS DE TODOS LOS PARTIDOS
-SERVIDOR CERRADO
-PUNTOS ENVIADOS CORRECTAMENTE AL PARTIDO
-FIN DE CONEXION CON EL PARTIDO
-FIN DE CONEXION ENTRE PARTIDOS
-PUNTOS COMPLETOS:
{"cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1":[[9997,2640083681790],
[3042,2640017038980]],"2f92bfab6e8f5a26f4914ba7aad3d90176786fa8e62d2b11acc5b215a972eca3acfd5f9ddd570a67807ba8f139639102888d500cffc834317c6580ca46c2d63efbdce640cb40b6061305a09fd54a695b39be9322b6f185bd8729fc1697c97fdd74511a0fc82cd0fb8a07bcc957d46a3d17a9d9778d301c6df996a38ef76ec410":[[9488,96943197],[1059,15822501]]}
Resultado
--------------------------------
{ Amarillo: 1, Verd: 1 }
```

Figura 10. Log del partido Amarillo.

Podemos observar que los parámetros RSA para los partidos son los mismos que para los usuarios. Además, vemos cómo se inician los dos servidores que se comentan en la sección de implementación {4.6} escuchando en el puerto 2000 y 3000 al igual que la IP del partido y la lista de IPs de todos los partidos.

Nótese que en este caso las identidades de los usuarios no se muestran cuando se conecta un usuario y así debe ser para que los partidos no puedan relacionar la dirección del voto con los electores.

Aclarado este punto, podemos ver cómo hay dos usuarios a los que se les ignora la petición puesto que su certificado es Error. Estos usuarios son el usuario *"44444444D"* y el segundo intento de votación del usuario "*11111111A"*. Aparte de estos dos usuarios, podemos ver que hay otros dos a los que sí que se les verifica la firma, y, al ser esta correcta se guarda el punto que han enviado en el diccionario de puntos.

Una vez ha terminado el proceso de elección, podemos ver como los partidos empiezan a comunicarse entre ellos y guardan en sus diccionarios todos los puntos que van recibiendo. Una vez finalizada la reconstrucción del conjunto de puntos, empieza la fase de recuperación de los votos mediante la interpolación de Lagrange y se empiezan a contar los votos recuperados. Esto da como resultado el diccionario que vemos en la Figura 8.

El resultado ha sido el esperado puesto que el partido *"Amarillo"* ha recibido un voto (El del elector *"11111111A"*) y el partido *"Verde"* ha recibido un voto (El del elector *"22222222B"*).

6.2.6 Partido Verde

El resultado del partido *"Verde"* va a ser el mismo que el del partido *"Amarillo"* pero cambiando los puntos que recibe del usuario. Al final, quedará el mismo diccionario que en el otro partido, pero cambiando el orden de los puntos. Aclarado esto, vamos a ver la salida que produce el partido *"Verde"* aunque no hay nada que comentar puesto que todo lo que había que detallar ya se ha explicado en el punto anterior.

```
-IP PROPIA: 192.168.16.3

-MODULO:
158091116696649633471988322770088590235531744390453768849995945521951178821999122962276541512230032734731394982462590313006831627577907231881351661045175483620515445156562907504992939719061323343851847408871777992745612703501600882663923141457968929560408150741172100231933679164575608410076076981511792733949
-EXPONENTE PUBLICO: 65537

-SERVIDOR TCP INICIADO EN EL PUERTO 2000
-SERVIDOR TCP INICIADO EN EL PUERTO 3000

-LISTA DE IP: 192.168.16.3,192.168.16.4

-USUARIO CONECTADO
-DATOS RECIBIDOS: 2912;2640016163144;297e8dc50f6a3cb898b044e34e28b5b35b6b83ef388daabe9c7f0ca63b8d25cf;Error
-USUARIO DESCONECTADO

-USUARIO CONECTADO
-DATOS RECIBIDOS:
3042;2640017038980;4374ff9964febe2d95e6880d06f5e2d9a66d9269a5ebefcdcc1ab3e4e178b993;cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1
-VERIFICACION: true
-PUNTOS:
{"cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1":[[3042,2640017038980]]}
-USUARIO DESCONECTADO

-USUARIO CONECTADO
-DATOS RECIBIDOS:
1059;15822501;aed50e19b2a31762559e0b04557fe2695ad3217a6063307497d1a34664530b3f;2f92bfab6e8f5a26f4914ba7aad3d90176786fa8e62d2b11acc5b215a972eca3acfd5f9ddd570a67807ba8f139639102888d500cffc834317c6580ca46c2d63efbdce640cb40b6061305a09fd54a695b39be9322b6f185bd8729fc1697c97fdd74511a0fc82cd0fb8a07bcc957d46a3d17a9d9778d301c6df996a38ef76ec410
-VERIFICACION: true
-PUNTOS:
{"cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1":
[[3042,2640017038980]],"2f92bfab6e8f5a26f4914ba7aad3d90176786fa8e62d2b11acc5b215a972eca3acfd5f9ddd570a67807ba8f139639102888d500cffc834317c6580ca46c2d63efbdce640cb40b6061305a09fd54a695b39be9322b6f185bd8729fc1697c97fdd745111a0fc82cd0fb8a07bcc957d46a3d17a9d9778d301c6df996a38ef76ec410":[[1059,15822501]]}
-USUARIO DESCONECTADO

-USUARIO CONECTADO
-DATOS RECIBIDOS: 2337;2639989706385;1a6b076cc8a4eb5803d8cdcdaba00189ab8c20837b0fa961b2a30bd916795b73;Error
-USUARIO DESCONECTADO

-SERVIDOR CERRADO
-PUNTOS ENVIADOS CORRECTAMENTE AL PARTIDO
-FIN DE CONEXION CON EL PARTIDO
-DATOS RECIBIDOS DE TODOS LOS PARTIDOS
-FIN DE CONEXION ENTRE PARTIDOS
-PUNTOS COMPLETOS:
{"cb8fadc26ad60e98dd981edc6f522dab71ff7c13d17575257cf8429827a5e93356b4861c20aff20d37e942f1b06e1c352f248913ff8e564cf630d78445157e72017624ccc3a8553582f4bb73382743cfaeb76e20758135f2db6ea70184538787dc526c3cb9e20af5cd99aef9371894c099f2f963f7f82fce093868d5cd7cc7b1":[[3042,2640017038980],
[9997,2640083681790]],"2f92bfab6e8f5a26f4914ba7aad3d90176786fa8e62d2b11acc5b215a972eca3acfd5f9ddd570a67807ba8f139639102888d500cffc834317c6580ca46c2d63efbdce640cb40b6061305a09fd54a695b39be9322b6f185bd8729fc1697c97fdd74511a0fc82cd0fb8a07bcc957d46a3d17a9d9778d301c6df996a38ef76ec410":[[1059,15822501],[9488,96943197]]}
Resultado
--------------------------------
{ Amarillo: 1, Verd: 1 }
```

Figura 11. Log del partido Verde.

7. Conclusión y trabajo futuro

En conclusión, se ha implementado una prueba de concepto que sirve de base para la evolución del protocolo. El esquema funciona de manera precisa y eficiente tal y como hemos podido comprobar a lo largo de las pruebas realizadas. Este es el primer paso para disponer de una aplicación implementada en un entorno real y, al ser una implementación distribuida, esto nos permite avanzar hasta un entorno de preproducción en el que se prepare el esquema y se hagan pruebas de eficiencia y requisitos para después usarlo en un entorno de producción.

Los conocimientos que han sido útiles para la realización del TFG han sido los siguientes:

Criptografía. Para el entendimiento teórico del esquema y el uso de todas las primitivas criptográficas como el concepto de seguridad perfecta, ecuaciones modulares y algoritmos como RSA, funciones hash.

Tecnología de sistemas de información en la red. Ha aportado los conocimientos necesarios para la implementación en Node.js (asincronía, creación de conexiones, módulos...) y el despliegue en Docker (creación de imágenes, contenedores y servicios).

Gestión de proyectos. Ha ayudado en la planificación del proyecto aportando los conocimientos necesarios para preparar el proyecto y realizar el diagrama de Gantt correspondiente.

Como trabajo futuro hay diferentes cosas en las que podemos mejorar este proyecto. En lo que sigue de sección, se comentarán diferentes avances a tener en cuenta en esta implementación.

Una posible mejora sería el acotar el nombre de los partidos. Así pues, tendríamos una lista con todos los partidos y si el voto realizado por un elector no coincide con el nombre de ninguno de los partidos que están en la lista se consideraría voto nulo. También podríamos hacerlo mediante una lista de selección en lugar de elegir el nombre del partido y así pues el entero que representa este partido podría ser su posición en la lista.

También deben hacerse cambios en cuanto a la parte del elector para poder usarse en un entorno real. Para que el elector votase más fácilmente se podría implementar una interfaz gráfica en la que la elección del partido sea visual y se puedan ver los boletines públicos para verificar su voto. En cuanto al despliegue, en este caso se han creado contenedores para los diferentes usuarios que han votado, pero en un entorno de producción esto no debe realizarse de esta manera. Tanto la autoridad identificadora como los interventores serían desplegados mediante docker-compose en una misma máquina y el docker-compose.yml correspondiente dejaría públicos sus puertos de comunicación con otros componentes (mediante la cláusula "ports:" correspondiente en cada "service"). Los electores votarían mediante una aplicación de escritorio o portal web y deberían conocer la dirección IP del host donde se hayan desplegado todos los

demás componentes, así como los puertos en los que escuchan (que siempre serían fijos, y estarían establecidos en el docker-compose.yml correspondiente). Esto no añadiría complejidad a la implementación, solo habría que eliminar a los electores del docker-compose y modificar ligeramente su script para recibir la información que sea necesaria para el proceso de votación.

El código fuente de este proyecto esta disponible en un repositorio público al que cualquiera tendrá acceso y podrá aportar sus mejoras. [33]

8. Referencias

Referencias

[1] T. Kohno, A. Stubblefield, A. Rubin y D. Wallach, «Analysis of an electronic voting system,» de *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, Berkeley, CA, USA, 2004.

[2] A. Larriba y D. López, «SUVS: Secure Unencrypted Voting Scheme,» *Informatica,* vol. 33, nº 4, pp. 749-769, 2022.

[3] A. Shamir, «How to share a secret,» *Communications of the ACM,* vol. 22, pp. 612-613, 1979.

[4] R. Rivest, «The ThreeBallot Voting System,» 2006.

[5] R. Rivest y W. Smith, «Three voting protocols: ThreeBallot, VAV, and Twin,» *Electronic Voting Technology,* 2007.

[6] D. Chaum, «Blind Signatures for Untraceable Payments,» *Advances in Cryptology,* pp. 199-203, 1983.

[7] C.-T. Li, M.-S. Hwang y Y.-C. Lai, «A Verifiable Electronic Voting Scheme over the Internet,» de *2009 Sixth International Conference on Information Technology: New Generations*, Las Vegas, NV, USA, 2009.

[8] T. Nguyen y T. Dang, «Enhanced security in internet voting protocol using blind signature and dynamic ballots,» *Electronic Commerce Research,* vol. 13, nº 3, pp. 257-272, 2013.

[9] A. Larriba, J. Sempere y D. López, «A two authorities electronic vote scheme,» *Computers & Security,* vol. 97, p. 101940, 2020.

[10] A. Thi y T. Dang, «Enhanced security in internet voting protocol using blind signature and dynamic ballots,» *Electronic Commerce Research,* vol. 13, nº 3, pp. 257-272, 2013.

[11] A. Aziz, «Coercion-Resistant E-Voting Scheme with Blind Signatures,» de *Cybersecurity and Cyberforensics Conference*, Melbourne, Autralia, 2019.

[12] J. Cruz y Y. Kaji, «E-voting system based on the bitcoin protocol and blind signatures,» *IPSJ Transactions on Mathematical Modeling and Its Applications,* vol. 10, nº 1, pp. 14-22, 2017.

[13] R. Cramer, R. Gennaro y B. Schoenmakers, «A secure and optimally efficient multi-authority election scheme,» *European Transactions on Telecommunications,* vol. 8, nº 5, pp. 481-490, 1997.

[14] T. ElGamal, «A public key cryptosystem and a signature scheme based on discrete logarithms,» *IEEE Transactions on Information Theory,* vol. 31, nº 4, p. 469–472, 1985.

[15] X. Yang, X. Yi, C. Ryan, R. Van Schyndel, F. Han, S. Nepal y A. Song, «A Verifiable Ranked Choice Internet Voting System,» de *Web Information Systems Engineering – WISE 2017 – 18th International Conference, Proceedings, Part II*, Puschino, Russia, 2017.

[16] X. Yang, X. Yi, S. Nepal, A. Kelarev y F. Han, «A Secure Verifiable Ranked Choice Online Voting System Based on Homomorphic,» *IEEE Access,* vol. 6, pp. 20506-20519, 2018.

[17] J. Tornos, J. Salazar, J. Piles, J. Saldana, L. Casadesus, J. Ruíz-Mas y J. Fernández-Navajas, «An eVoting System Based on Ring Signatures,» *Network Protocols & Algorithms,* vol. 6, nº 2, pp. 38-54, 2014.

[18] G. Chen, C. Wu, W. Han, X. Chen, H. Lee y K. Kim, «A new receipt-free voting scheme based on linkable ring signature for designated verifiers,» de *International Conference on Embedded Software*, 2008.

[19] X. Yang, X. Yi, S. Nepal, A. Kelarev y F. Han, «Blockchain voting: publicly verifiable online voting protocol without trusted tallying authorities,» *Future Generation Computer Systems,* vol. 112, pp. 859-874, 2020.

[20] S. Gao, D. Zheng, R. Guo, C. Jing y C. Hu, «An Anti-Quantum E-Voting Protocol in Blockchain With Audit Function,» *IEEE Acces,* vol. 7, pp. 115304-115316, 2019.

[21] H. Niederreiter, «A Public-Key Cryptosystem based on Shift Register Sequences,» de *Workshop on the Theory and Application of of Cryptographic Techniques*, 1985.

[22] A. Larriba, A. Cerdà i Cucó, J. Sempere y D. López, «Distributed Trust, a Blockchain Election Scheme,» *Informatica,* vol. 32, nº 2, p. 321–355, 2021.

[23] C. Shannon, «Communication theory of secrecy systems,» *Bell Labs Technical Journal,* vol. 28, nº 4, pp. 656-715.

[24] «Web oficial Node.js,» [En línea]. Available: https://nodejs.org/es.

[25] «Acerca de Node.js,» [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-node-js.

[26] «Modulo net,» [En línea]. Available: https://nodejs.org/api/net.html.

[27] «Modulo crypto,» [En línea]. Available: https://nodejs.org/api/crypto.html.

[28] «Modulo fs,» [En línea]. Available: https://nodejs.org/api/fs.html#file-system.

[29] «Modulo big-int,» [En línea]. Available: https://www.npmjs.com/package/big-integer.

[30] «Modulo os,» [En línea]. Available: https://nodejs.org/api/os.html.

[31] M. Jones, «JSON WEB KEY JWK,» 2015.

[32] «Sitio oficial de Docker,» [En línea]. Available: https://www.docker.com/.

[33] «Repositorio del proyecto,» [En línea]. Available: https://github.com/OscRosIba/SUVSProyect.

9. Anexo

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**x**
ODS 2. **Hambre cero.**				**x**
ODS 3. **Salud y bienestar.**		**x**		
ODS 4. **Educación de calidad.**				**x**
ODS 5. **Igualdad de género.**				**x**
ODS 6. **Agua limpia y saneamiento.**				**x**
ODS 7. **Energía asequible y no contaminante.**			**x**	
ODS 8. **Trabajo decente y crecimiento económico.**				**x**
ODS 9. **Industria, innovación e infraestructuras.**				**x**
ODS 10. **Reducción de las desigualdades**.				**x**
ODS 11. **Ciudades y comunidades sostenibles**.			**x**	
ODS 12. **Producción y consumo responsables.**				**x**
ODS 13. **Acción por el clima.**	**x**			
ODS 14. **Vida submarina.**				**x**
ODS 15. **Vida de ecosistemas terrestres.**				**x**
ODS 16. **Paz, justicia e instituciones sólidas.**	**x**			
ODS 17. **Alianzas para lograr objetivos.**	**x**			

Este trabajo se puede relacionar con diferentes ODS.

Salud y bienestar (ODS 3). La implementación de un protocolo de votación electrónica puede ayudar en cuanto al bienestar se refiere puesto que facilita la vida de la gente permitiendo a los electores realizar su voto desde casa sin necesidad de acudir al colegio electoral.

Energía asequible y no contaminante (ODS 4). Al eliminar la necesidad de acudir a votar al colegio electoral, esto permite que se reduzca el uso de vehículos y por tanto ayuda a reducir la contaminación por combustibles. Al ser este un proceso que se realiza pocas veces y que la gente suele tener próximo el lugar de votación se considera que el grado de relación es bajo.

Ciudades y comunidades sostenibles (ODS 11). Por el mismo motivo que antes, la reducción del uso de vehículos para acudir al colegio electoral ayuda a crear ciudades y comunidades más sostenibles. El grado de relación de este ODS también se considera bajo.

Acción por el clima (ODS 13). El hecho de reducir todo el papel usado y todo el coste en logística en el proceso de elección hace de este un planeta más verde. La cantidad de dinero y recursos necesarios en un proceso de elección es bastante grande por lo que, si eliminamos todo esto, permitiendo que sea posible unas elecciones en las que solo serían necesarios unos pocos dispositivos electrónicos además de los dispositivos que seguramente ya poseen los electores, podríamos reducir por completo todo el gasto en recursos físicos y de logística.

Paz, justicia e instituciones sólidas. (ODS 16). La creación de un esquema de votación electrónica seguro y eficiente nos permite dar solidez a las instituciones públicas y aumentar el grado de confianza de la gente en los sistemas políticos.

Alianzas para lograr objetivos. (ODS 17). La alianza de todos los interventores en un proceso de elección utilizando este esquema es necesaria por lo que se considera que es necesario que todas las partes cooperan por lograr el bien común.