Informe eSAMCid sobre el voto electrónico

Enero de 2017

Resumen

En este documento se hace una presentación del voto electrónico. El objetivo de este documento es dar a conocer con bastante detalle en qué consiste un sistema de votación electrónica. Además de mostrar su utilidad, se exponen los requisitos de seguridad que se exigen, así como la complejidad que conllevan. Este documento también presenta de forma exhaustiva los protocolos de votación electrónica de los que se dispone en la actualidad explicando las características principales de cada uno de ellos, así como las mejoras o aportaciones que introducen. En la última sección se analiza en profundidad el tema de lo que se denomina verificabilidad, este aspecto es fundamental para que cualquier usuario del sistema se sienta seguro con todo el proceso de votación, no solo porque esté basado en protocolos en los que hay que confiar, sino porque puede verificar cualquiera de los pasos involucrados en el proceso de la votación.

El texto está escrito sin entrar en muchas definiciones ni descripciones de los componentes criptográficos que requiere cualquier sistema de votación electrónica. Se ha decidido hacerlo así para permitir una lectura más fluida. Sin embargo, es cierto que esta elección puede, en ocasiones, dificultar la comprensión, por ello al final del documento se listan los principales conceptos de criptografía que aparecen en el texto, acompañados del enlace de wikipedia en el que están descritos con el suficiente detalle para comprender completamente su uso en los protocolos que lo requieran.

Subvencionado por el Ministerio Español de Economía y Universidades mediante el projecto MTM2013-41426-R.

Índice

1. Introducción **1**
1.1. El voto electrónico 1
1.2. El voto por Internet 1
1.3. El voto por correo 2
1.4. Requisitos del voto por internet 2
1.5. Solución básica para voto electrónico 5
1.6. Un ejemplo de complejidad: gestión de claves 7

2. Protocolos criptográficos para sistemas de voto electrónico **10**
2.1. Protocolos de Pollsterless o papeletas precifradas . . . 10
2.2. Modelo de las dos agencias 12
2.3. Protocolos de recuento homomórfico 14
2.4. Protocolos de Mezcla o Mixing 17
2.4.1. Mixnets de descifrado 18
2.4.2. Mixnet de recifrado 18

3. Nuevas variantes **20**
3.1. Mixnets híbridas 20
3.2. Vector-Ballot 21
3.3. Mixing de credenciales 23

4. Verificabilidad **26**
4.1. Problemática: no son personas, es software 26
4.2. Herramientas criptográficas de verificación 27
4.3. Verificación cast-as-intended 29
4.3.1. Desafiar al cliente de votación 29
4.3.2. Los códigos de retorno 32
4.4. Verificación counted-as-cast 35
4.4.1. Esquemas de recuento homomórfico 35
4.4.2. Mixnets 36
4.5. Recibos de votación: verificación counted-as-cast o recorded-as-cast 39

5. Referencias básicas sobre criptografía **42**

Referencias **43**

1. Introducción

1.1. El voto electrónico

Hace bastantes años que se utilizan sistemas de votación electrónica. Básicamente, la necesidad de facilitar el proceso del recuento de votos, o el voto a distancia, por ejemplo para residentes en el extranjero o personas con movilidad reducida (lo que puede aumentar el índice de participación), la reducción de costes a largo plazo o la capacidad de mostrar las papeletas en diferentes idiomas, han impulsado el uso de herramientas electrónicas en este campo.

En 1984, el estado de Illinois empezó a probar sistemas electrónicos de recuento de votos, y hacia el 2000 ya se habían implementado sistemas de votación electrónica en Bélgica, Brasil, Finlandia y Alemania, entre otros [10].

Quizás en España no se ha visto una gran necesidad de aplicar herramientas electrónicas en los procesos electorales ya que, en general, nuestras elecciones son sencillas y no muy frecuentes. Hemos de pensar que hay países en los que se permite votar con listas abiertas escogiendo candidatos de diferentes partidos, usando el voto preferencial y poniendo pesos a los candidatos, o se realizan votaciones para varias cuestiones a la vez, o se someten a referéndum muchas cuestiones que afectan a los ciudadanos, etc. Todo ello hace muy complejo el proceso de gestión de las papeletas así como el del recuento.

1.2. El voto por Internet

Con el nacimiento de Internet nos hemos ido convirtiendo en individuos cada vez más conectados y acostumbrados a realizar gestiones de forma remota por Internet. La mayoría de nosotros tiene presente que la flexibilidad que nos proporciona Internet (creación de múltiples identidades digitales, ubicuidad de las comunicaciones, publicación de información nuestra en diferentes ámbitos), conlleva riesgos en la red (ataques de suplantación de identidad, observación o interceptación de las comunicaciones, etc.). Sin embargo, esto no impide que queramos hacer operaciones delicadas de forma remota, como gestiones bancarias, declaraciones de renta, compra de billetes de avión, etc. Por tanto, la implantación del voto por Internet, donde los votantes pueden votar desde su casa o su trabajo, usando un ordenador, un teléfono móvil o una tableta y una conexión a Internet, surge de forma natural.

El voto por Internet, no sólo puede contribuir a aumentar los índices de participación porque los votantes pueden elegir desde donde emitir su voto, sino porque facilita la participación de algunos colectivos que no pueden desplazarse al colegio electoral el día de la votación, bien por estar fuera de su país o región (militares desplazados en una misión, pescadores embarcados) bien por tener problemas de movilidad.

1.3. El voto por correo

Es sabido que para estos colectivos existe el voto por correo: el votante, antes de las elecciones, solicita esta modalidad de voto para que le envíen las papeletas de voto a la dirección indicada, desde donde enviará al colegio electoral el sobre con la papeleta con las opciones de voto elegidas. No obstante, esta solución no siempre funciona. Las listas de candidatos han de estar definidas y cerradas tiempo antes de la elección para tener tiempo de enviar las papeletas a los votantes desplazados y que éstos tengan tiempo de reenviarlas por correo postal, pero muchas veces estos tiempos son muy ajustados: recordemos los problemas del voto postal en las elecciones del 20N [1], cuando el plazo para el voto por correo finalizaba y los votantes todavía no habían recibido las papeletas. Si pensamos en militares desplazados a zonas de conflicto, podemos imaginar que el sistema de correo postal no funcionará demasiado bien. Justamente este problema motivó las leyes relacionadas con el Uniformed and Overseas Citizens Absentee Voting Act (UOCAVA) y Military and Overseas Voter Empowerment Act (MOVE) en Estados Unidos [7, 8], que obligan a los estados a proporcionar a militares y ciudadanos residentes fuera de Estados Unidos medios electrónicos para solicitar y recibir sus papeletas de voto. Por tanto, el voto postal no siempre puede proporcionar un servicio suficiente a los votantes ausentes.

1.4. Requisitos del voto por internet

El voto por Internet es un campo de estudio muy interesante ya que es un proceso con unos requisitos de seguridad muy fuertes que no han de impedir que el sistema sea de fácil uso, se ejecuta en un entorno que se considera inseguro y las contramedidas aplicables son limitadas.

Los requisitos de seguridad de los procesos electorales, tanto si se realizan de forma tradicional como de forma electrónica, son los siguientes:

- Autenticación de los votantes: Se ha de asegurar que los votos son

emitidos por votantes válidos para esta elección y que sólo se cuenta un voto por votante.

- Privacidad de los votantes: Al mismo tiempo que ha de ser posible identificar a los votantes para asegurar que son votantes válidos, ha de ser imposible relacionar la identidad de un votante con el contenido del voto que ha emitido.
- Precisión de los resultados de la elección: Hay que garantizar que los resultados de la elección se corresponden con los votos emitidos por los votantes, de forma que no ha de ser posible que alguien pueda modificar o eliminar votos válidos o añadir votos falsos en nombre de votantes que se hayan abstenido.
- Secreto de los resultados intermedios: Los resultados intermedios, en el sentido del resultado de contar los votos emitidos antes de que haya acabado el período de votación, ha de ser secreto hasta el fin de dicho período, para evitar influenciar sobre los votantes que todavía no hayan emitido su voto.
- Verificabilidad: Un votante ha de poder verificar de forma independiente que el voto emitido representa la opción escogida, que ha sido incluido en el recuento final y que el proceso del recuento ha sido correcto. Un auditor ha de poder verificar de forma independiente que todos los votos emitidos por votantes válidos han sido incluidos en el recuento.
- No coacción: un votante no ha de poder demostrar a un tercero cómo ha votado, para evitar la compraventa de votos o la posibilidad de que los votantes sean coaccionados.

Si nos fijamos en estos requisitos, algunos parecen contradictorios: queremos poder identificar a un votante que ha emitido un voto sin poderlos relacionar, y queremos que el votante pueda verificar su voto sin que sea posible mostrárselo a una tercera persona. Son este tipo de requisitos con exigencias contrarias los que hacen que los protocolos criptográficos enfocados a satisfacerlos no sean triviales, como se verá en las siguientes secciones.

En relación con el entorno de ejecución de este proceso, suponemos que trabajamos con el siguiente esquema (ver figura 1): el votante usa un ordenador o un dispositivo con capacidad criptográfica (hoy en día ya podemos hablar de tabletas o teléfonos inteligentes) para emitir su voto, que se envía

a través de Internet a un servidor de voto remoto donde se almacena. Al final de la elección se procede al recuento de los votos recibidos y almacenados en el servidor de voto para obtener el resultado.

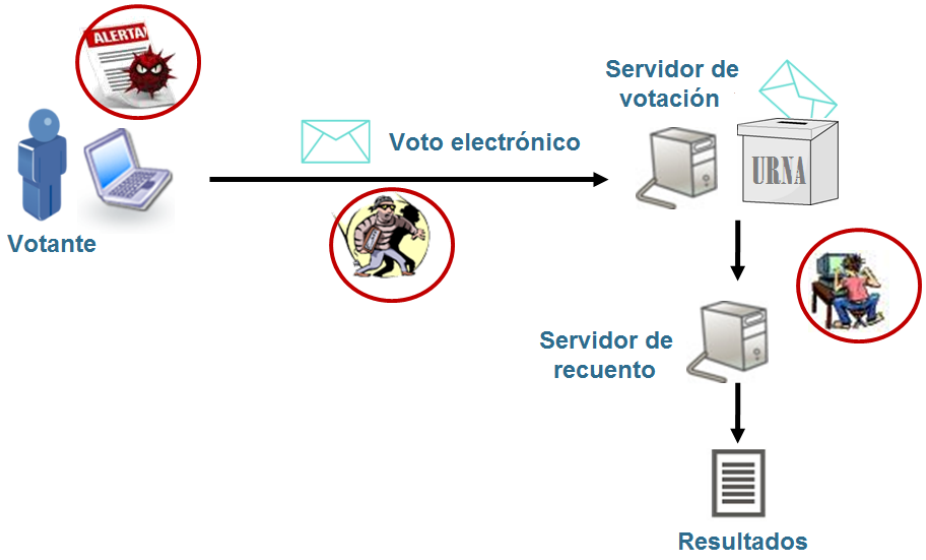

Figura 1: Escenario típico de voto electrónico remoto y puntos de riesgo.

De cara a las medidas necesarias para cumplir con los requisitos de seguridad, se supone lo siguiente:

- Es posible que el dispositivo utilizado por el votante conozca las opciones de voto elegidas por dicho votante y que las pueda modificar antes de enviarlas al servidor de voto remoto. También es posible que otras personas tengan acceso a ese mismo dispositivo.
- Es posible que alguien pueda observar la comunicación entre el dispositivo del votante y el servidor de voto ya que se hará a través de una red pública (Internet). Aunque TLS es un protocolo pensado para aportar seguridad a las comunicaciones de esta naturaleza, los numerosos ataques y las vulnerabilidades [46] del sistema de Infraestructura de Clave Pública o PKI (del inglés Public Key Infrastructure) sobre el que se sustenta su seguridad, motivan que no supongamos privacidad de la comunicación incluso usando TLS.
- El servidor de voto no es un sistema cerrado: además de tener la capacidad de recibir votos por Internet, un administrador del sistema o un operario puede tener acceso y cambiar los votos en el servidor.

- El proceso de recuento electrónico de los votos se ejecuta en una máquina diferente o en el mismo servidor de voto, donde un administrador del sistema o un operario puede tener acceso y hacer algún cambio en el resultado.

Ben Adida, en su tesis [10], explica que buena parte de la opinión pública compara el voto electrónico con otros sistemas más complejos, como puede ser el control de aviones o el procesado de transacciones bancarias. En un primer momento parece cierto que si somos capaces de diseñar sistemas electrónicos para realizar estas tareas, no tendría que haber impedimento para utilizar medios electrónicos en los procesos electorales. Según explica, estas analogías cometen tres errores muy notables: Primero, los interesados en manipular una elección pueden ser mucho más poderosos de lo que a menudo imaginamos. Segundo, el modelo de seguridad para aviones y bancos es menos exigente que para el voto electrónico, principalmente por esos requisitos aparentemente contradictorios que exponíamos anteriormente. Tercero, la detección de fallos y la capacidad de recuperación son cosas bien definidas en el caso de aviones y bancos. En el caso de los procesos electorales no está claro si todas las cosas que pueden ir mal son detectables, y la recuperación suele ser imposible (habría que repetir las elecciones). Así pues, el voto electrónico está considerado un campo de aplicación de la seguridad con un escenario muy complejo.

Intuitivamente, muchas de las medidas que pueden aplicarse para resolver todos los problemas que conlleva el proceso de votación electrónica están relacionadas con la criptografía. Herramientas criptográficas como el cifrado o la firma digital permiten proteger la privacidad del votante, o asegurar la integridad del voto emitido. No obstante, estas medidas no bastan, va a ser necesario considerar protocolos criptográficos más avanzados, de los cuales hay diversas propuestas, aunque aún no se ha encontrado la gran solución que satisfaga todos los requisitos de una manera conveniente, como se verá a lo largo de este documento. Es por ello que esta problemática recibe tanto interés por parte de los investigadores en criptografía y seguridad.

1.5. Solución básica para voto electrónico

Como ya se ha dicho, cifrar un voto y firmarlo no basta para cumplir los requisitos de seguridad de los procesos de voto. Imaginemos el siguiente esquema (ver figura 2):

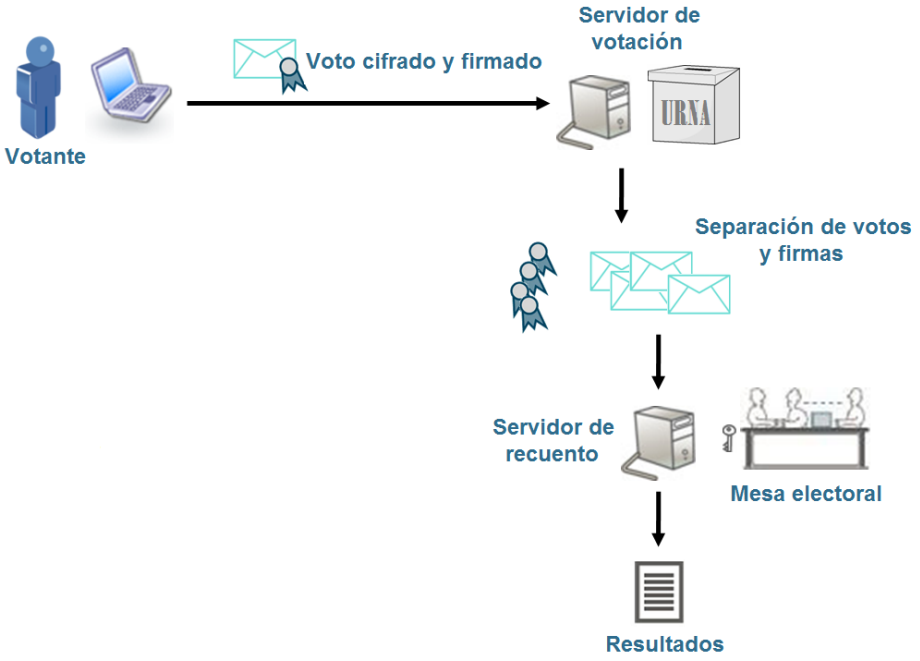

Figura 2: Solución básica con cifrado y firma de voto antes de enviar.

El votante utiliza un cliente de voto, que es un software que se ejecuta en su ordenador o terminal de voto (por ejemplo, un teléfono inteligente) para escoger las opciones de voto, que se cifran y firman digitalmente antes de enviarse a un servidor de votación remoto, donde se verifica que la firma ha sido realizada por un votante válido y se almacenan. Una vez acabada la elección, se separan los votos de sus firmas, la mesa electoral descifra los votos, se calculan los resultados de la elección y se publican.

La firma digital de los votos cifrados permite la verificación de la integridad del voto una vez es recibido en el servidor de voto, para comprobar que no ha sido manipulado durante su transmisión. También permite verificar la validez del votante en esa elección (es decir si la persona está autorizada para votar en la elección), ya que la firma suele ir ligada a algún tipo de certificado digital que indica la identidad del firmante. Antes de descifrar los votos, éstos se separan de sus firmas, para evitar la relación entre el contenido de un voto en claro y la identidad del votante que lo ha emitido. Este método es muy parecido al del voto por correo, donde el votante envía el voto dentro de un sobre (análogo al cifrado) y pone éste dentro de un sobre firmado. Una vez comprobada la firma comparándola con un registro

previo, el sobre interior (ya anónimo) se pone en una urna con más sobres, que se abrirán (análogo al descifrado) en el momento del recuento.

Aunque este esquema parezca suficientemente seguro para proteger el secreto de voto, si alguien es capaz de observar el orden en que los votos cifrados y firmados se reciben en el servidor y el orden en que son descifrados, la privacidad del votante quedaría comprometida, relacionando el contenido de un voto con un votante específico. Además, este sistema no permite verificar que los votos no se modifican una vez recibidos y almacenados en el servidor de voto (que hace de urna) hasta que se descifran. Así pues, las medidas de firmar y cifrar no bastan por sí solas.

1.6. Un ejemplo de complejidad: gestión de claves

Un aspecto de gran relevancia es el impacto que tienen las medidas criptográficas en la complejidad del sistema. Utilizaremos como ejemplo la gestión de claves. Aunque el esquema es muy sencillo necesita unos ciertos procesos de gestión de claves debido a las herramientas criptográficas que usa, además de la configuración del proceso electoral en sí mismo:

Primero, notemos que hemos de utilizar un algoritmo de clave pública para cifrar los votos. En caso de utilizar uno de clave privada, o bien cada votante usa una clave diferente compartida con la mesa electoral para poder cifrar su voto, con lo que la gestión de todas esas claves sería insostenible, o todos los votantes usan la misma clave, con lo que un votante que interceptara el voto cifrado de otro votante podría descifrarlo. Entonces, será necesario, en algún momento previo al inicio de la fase de votación, generar una pareja de claves pública-privada de la elección, donde la clave pública será la que usen los votantes para cifrar sus votos.

Una práctica habitual para proteger la clave privada hasta el final del proceso, de manera que no se puedan descifrar los votos antes de que acabe la elección, consiste en partir la clave privada en trozos o shares, mediante un esquema de compartición de secretos [62] y repartirla entre los miembros de la mesa electoral. Una vez finalizada la fase de votación, los miembros de la mesa electoral dan sus fragmentos para reconstruir la clave privada y poder descifrar los votos. Con el objetivo de prevenir que la ausencia de un miembro de la mesa (o que se niegue a participar en el proceso) impida la reconstrucción de la clave, normalmente se usa un esquema de compartición de secretos de umbral [62] donde se requiere la participación de un determinado número de miembros de la mesa electoral (por ejemplo la mayoría), para reconstruir el secreto.

En un esquema clásico de compartición de secretos, el secreto existe en un primer momento y luego se reparte en fragmentos, por tanto todavía existe el peligro de que alguien intercepte el secreto antes de ser repartido. Existen técnicas [13, 17, 24, 25, 30, 32] que permiten que cada miembro de la mesa genere por su cuenta un secreto que será su fragmento de la clave privada y muestre a los demás su parte correspondiente de la clave pública, la clave pública de la elección se forma a partir de esos fragmentos de clave pública de cada miembro de la mesa y la clave privada completa no existe en ningún momento. Este proceso es, generalmente, más costoso que el anterior, especialmente cuando el algoritmo para el que se están generando las claves es RSA [2, 47, 58]. Esto es debido a que hay que ejecutar varias rondas del protocolo para encontrar un módulo RSA que sea producto de dos primos (habrá que ejecutar test de primalidad sobre estos números que forman parte de la clave privada) y ha de ser posible generar claves de manera segura aun en presencia de participantes maliciosos en el protocolo. Ivan Damgaard [26] entre otros, ha estado trabajando en los últimos años en procesos que permitan acelerar este proceso. En [29] se propone una modificación de [30] para mejorar la eficiencia. En la mejora propuesta, el protocolo se ejecuta suponiendo que todo el mundo actúa de forma honesta, y en caso de encontrar un módulo producto de dos primos se hace una ronda de verificación para comprobar que los participantes no han mentido. De esta manera las rondas donde no se encuentra un módulo válido son mucho más rápidas y sólo cuando se encuentra una válida se comprueba si todo ha sido realizado de forma correcta siguiendo un protocolo más costoso.

Segundo, el votante ha de tener un par de claves propias para poder firmar digitalmente su voto. La firma digital del voto permite asegurar su integridad durante el envío del mismo al servidor de votación y hasta el momento del descifrado y recuento, pero también permite contar un solo voto por votante y asegurar que todos los votos contabilizados han sido emitidos por votantes que están en el censo electoral: la clave pública que se encuentra en el certificado digital, permite relacionar la firma con la identidad real de un votante. En nuestro país tenemos el eDNI que nos proporciona capacidades de autenticación y firma digital, pero hay otros países donde los votantes no tienen medios para realizar firmas digitales y tendrá que ser la plataforma de voto la que se encargue de proporcionar, de forma segura, estas claves de firma a los votantes, después de que éstos se hayan identificado. Por ejemplo, utilizando técnicas de key roaming: las claves de firma de los votantes están almacenadas en un servidor remoto (por ejemplo, el mismo servidor de voto o un servicio de autenticación). Cada clave está

protegida mediante un algoritmo de cifrado, por ejemplo simétrico o del tipo Password-Based Encryption (definición en PKCS5 y PKCS12 [3, 4]), con una clave secreta o password que conoce el votante, pero no el servicio que las almacena. De esta forma, después de identificarse como votante de la elección, el servicio proporciona al votante sus claves de firma, cifradas, que sólo el votante podrá abrir ya que es el único que conoce el valor secreto para descifrarlas.

Así pues, hemos visto que un esquema muy simple, que ni tan solo llega a cumplir los requisitos de seguridad de un proceso electoral, necesita de un proceso de gestión de claves que mantenga la seguridad del esquema y no penalice su eficiencia. Se puede deducir que, a medida que el esquema gane en complejidad, esta gestión de claves se hará más complicada si no se tiene en cuenta durante el diseño.

2. Protocolos criptográficos para sistemas de voto electrónico

A partir de la solución básica explicada en la sección 1.5 se han ido desarrollando protocolos criptográficos más complejos con el objetivo de cumplir con los requisitos de seguridad de los procesos electorales. Estos protocolos han sido diseñados para voto electrónico o se han desarrollado a partir de herramientas utilizadas en escenarios con requisitos de seguridad similares (por ejemplo, navegación anónima con sistemas como TOR [27]). Los protocolos que explicaremos se centran en proporcionar anonimato a los votantes (o secreto de voto) al mismo tiempo que cumplen con los requisitos de autenticación y precisión de los resultados. Estos protocolos se pueden dividir en cuatro tipos básicos:

2.1. Protocolos de Pollsterless o papeletas precifradas

Estos protocolos [45, 64] aplican medidas criptográficas principalmente en la fase de preelección, en la fase de diseño de las papeletas: a cada opción de voto se le asigna un código resultante de cifrar la opción. Las papeletas con los valores de los precifrados de cada opción de votación se envían a los votantes antes de la fase de votación (ver figura 3).

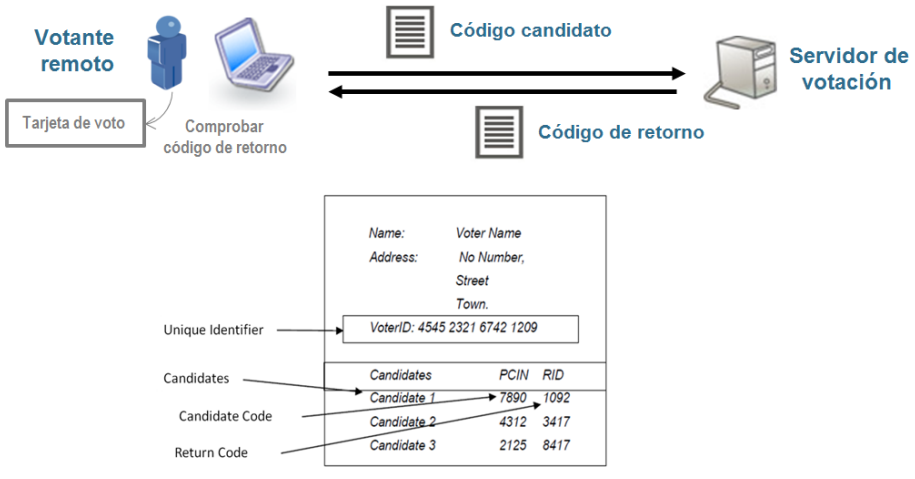

Figura 3: Protocolo de Pollsterless.

Estos métodos reciben el nombre de Pollsterless porque, a diferencia del

esquema de la sección 1.5, no necesitan que el terminal de voto tenga capacidad criptográfica para cifrar las opciones de voto del votante: en vez de que el terminal de voto cifre y firme el voto, el votante introduce el código del precifrado que aparece al lado de la opción deseada en la papeleta de voto que ha recibido, para que el terminal lo envíe al servidor remoto. La generación de los códigos y su distribución se hace de manera que sólo el votante conoce el valor de sus códigos y a qué opciones representan. Una ventaja de estos métodos es que se pueden usar para votar mediante el móvil , por ejemplo con un mensaje SMS.

El mecanismo de diseño de las papeletas es el siguiente:

- A cada partido o candidato se le asigna un código diferente por votante mediante el cifrado de la opción. Este código se puede generar por ejemplo usando una operación criptográfica de Hash [61] basada en la opción de voto y cifrándolo con una clave secreta relacionada con un identificador único de la papeleta.

- Opcionalmente, también se puede generar un segundo código, denominado código de retorno, diferente para cada candidato o partido. Este código se puede generar usando la misma función de Hash, pero encima del código anterior y usando una clave secreta que solo conoce el servidor de voto. El código de retorno se usa para verificar el registro correcto del voto, tal como se explica más adelante.

Antes del periodo de votación se envían por correo las papeletas asignadas de forma aleatoria a los votantes.

Iniciada la fase de votación, el votante accede a la aplicación de voto y emite su voto enviando la identificación única de la papeleta (CardID) y los códigos correspondientes a los candidatos escogidos (PCIN). El servidor recibe el voto y calcula los códigos de retorno (RID) a partir de los códigos recibidos (PCIN) y de su clave secreta. El votante recibe los códigos de retorno (RID) y comprueba si su valor coincide con los códigos de retorno de su papeleta asignados a las opciones escogidas. Así, el votante puede comprobar que su voto ha sido recibido y procesado correctamente por el servidor de voto, dado que un atacante no puede conocer por adelantado los códigos de retorno correspondientes a las opciones enviadas.

Finalmente, en la fase de recuento, se recupera la información de los candidatos relacionados con los códigos (PCIN) de los votos, usando la clave secreta asignada al identificador de la papeleta (CardID).

Desde el punto de vista de la privacidad, estos protocolos permiten:

- Emisión de votos anónimos: no están firmados por los votantes.
- Secreto de voto: el conocimiento de los códigos por parte del servidor de voto, o su interceptación, no permiten saber a qué candidato ha votado el votante, sin tener la papeleta.

El problema es que aún existe la posibilidad de romper la privacidad del votante si las papeletas con los valores precifrados son comprometidas. Por ejemplo, un tercero podría conocer la intención de voto de un votante interceptando los códigos del voto emitido y comparando los valores con los de la papeleta.

2.2. Modelo de las dos agencias

Estos protocolos [19, 31, 49] implementan las medidas criptográficas en la fase de votación y se centran en conseguir un canal de voto anónimo para que el votante pueda emitir su voto sin necesidad de revelar su identidad.

El modelo de las dos agencias (ver figura 4) se basa en la existencia de dos servicios completamente independientes: uno para identificar al votante y otro para emitir el voto.

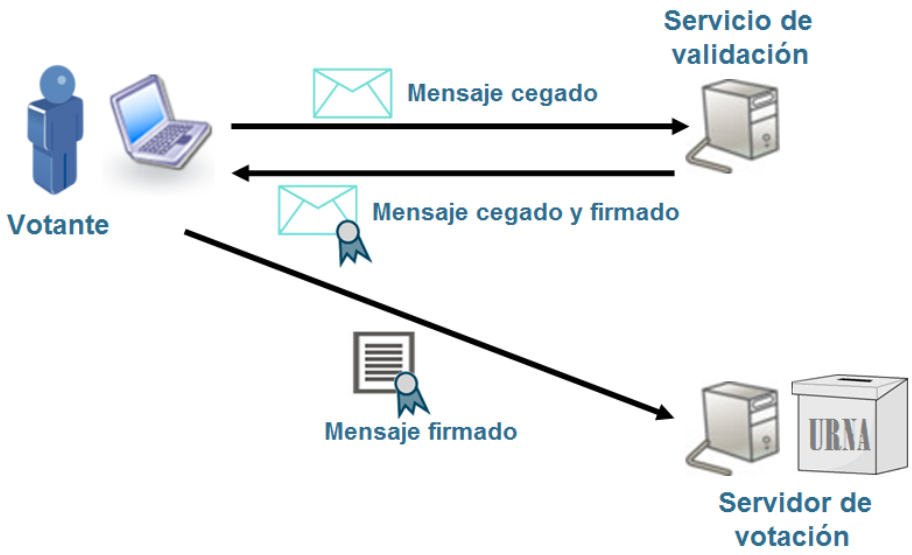

Figura 4: Modelo de las dos agencias.

- El Servicio Validador: identifica al votante, comprueba que éste pueda votar en la elección y permite que vote de forma anónima proporcionándole un *token* anónimo. Por token entendemos un mensaje de autorización que permite obtener permisos para acceder a otro servicio, dado que existe una garantía de que este mensaje ha sido emitido por el Servicio Validador (por ejemplo, una firma digital).

- El Servicio de Voto: recibe el voto cifrado del votante y el token anónimo que asegura que éste ha sido identificado por el Servicio Validador.

En estos esquemas, normalmente se utiliza un esquema de firma ciega [20], el Servicio Validador firma el voto que emitirá el votante sin saber exactamente qué está firmando (el contenido del voto). Este tipo de firma puede conseguirse gracias a ciertas propiedades matemáticas de algunos algoritmos como el RSA, por ejemplo, la maleabilidad [66]. Un algoritmo de cifrado con esta propiedad es aquel que al realizar una operación sobre el cifrado de un mensaje y descifrar el resultado se obtiene el mensaje original transformado. Por ejemplo, si $E(v)$ es el cifrado de un voto y lo operamos con un entero r, diremos que el algoritmo de cifrado es maleable si el descifrado $D(E(v) \cdot r)) = v * s$.
En un algoritmo de firma ciega basado en RSA, se multiplica el mensaje m por un entero r (llamado factor de ceguera) elevado a e (clave pública) módulo N (clave pública). El servicio de firma firma el mensaje cegado con su clave privada $(mr^e)^d = m^d r^{ed} = m^d r$. Ahora ya, quitando el factor de ceguera r queda el mensaje m firmado, sin que el servicio de firma lo conozca. Hay que puntualizar que en un esquema real de firma ciega, el valor a firmar no sería el mensaje m sino $H(m)$, donde H denota una función criptográfica unidireccional como por ejemplo una función de hash [61].
El proceso de voto es el siguiente:

- El votante cifra su voto utilizando la clave pública de la elección, lo ciega y lo envía al Servicio Validador junto con la información necesaria para que el Servicio Validador lo identifique (lo que usualmente se conoce como credenciales: usuario y contraseña, eDNI, etc.).

- El Servicio Validador recibe el mensaje cegado, comprueba que el votante que lo ha enviado puede votar en esta elección, y lo firma a ciegas, devolviendo esta firma al votante.

- El votante recibe la firma, elimina el factor de ceguera y obtiene la firma digital del Servicio Validador sobre el voto cifrado.

- El votante envía el voto cifrado y su firma al Servicio de Voto, que comprueba que la firma ha sido realizada por el Servicio Validador, y almacena el voto.

Cuando acaba la fase de votación, los votos se descifran con la clave privada de la elección y se cuentan para obtener los resultados.

Estos sistemas protegen la privacidad del votante mediante:

- La obtención por parte del votante de un voto firmado a ciegas (sin ver su valor real) por el Servicio de Validación.
- Evitando que el votante haya de identificarse en el servicio que recibe el voto.
- El cifrado de las opciones de voto antes de ser emitidas.

Hay que tener presente que la privacidad del votante depende principalmente de que las dos entidades no colaboren, ya que en caso contrario podrían compartir información (por ejemplo, direcciones IP de conexión) que permitirían relacionar los votos con los votantes.

También existe el riesgo de manipulación de la elección en caso de que el Servicio de Validación se vea comprometido: esta entidad puede generar de forma arbitraria votos cifrados y firmados que serían aceptados por el Servicio de Voto.

2.3. Protocolos de recuento homomórfico

Estos protocolos [16, 23] actúan principalmente en la fase de recuento. Utilizan propiedades homomórficas de ciertos sistemas de cifrado, en el sentido de que cierta operación entre los votos cifrados corresponde al cifrado del resultado de cierta operación entre los votos en claro.

Es decir, si tenemos dos votos v_1 y v_2 y sus correspondientes cifrados $E(v_1)$, $E(v_2)$, suponiendo $\cdot$ y $*$ dos operaciones algebraicas, un cifrado homomórfico se define como aquel que satisface:

$$E(v_1) * E(v_2) = E(v_1 \cdot v_2)$$

Según la operación de los votos sea la suma o el producto, se habla de homomórfico aditivo o multiplicativo.

El homomorfismo aditivo es el más utilizado en votación electrónica. Dado que permite obtener directamente la suma de los votos. No todos los algoritmos tienen estas propiedades, los más utilizados son ElGamal [28] o Paillier [50]. En ambos casos, el resultado de multiplicar los votos cifrados es la suma de votos cifrada.

$$E(v_1) \cdot E(v_2) = E(v_1 + v_2)$$

Entonces, sólo es necesario descifrar el resultado del producto de cifrados, para obtener la suma de los votos. Como se verá más adelante, los votos han de tener un formato numérico especial para obtener como resultado el número de veces que se ha seleccionado cada candidato u opción de voto por separado.

En el homomorfismo multiplicativo, el resultado de multiplicar los votos cifrados es el cifrado de la multiplicación de los votos.

$$E(v_1) \cdot E(v_2) = E(v_1 \cdot v_2)$$

Éste se usa menos en votación electrónica, ya que no se obtiene de forma directa la suma de los votos.

Dependiendo del homomorfismo que se utilice, el voto se representa de maneras diferentes. Por ejemplo, en el caso de homomorfismo multiplicativo se suele usar ElGamal en su versión exponencial: un elemento g de un grupo se usa para representar una opción de voto y se eleva a 1 o 0 según esa opción se elija o no. El valor resultante se cifra con el algoritmo de ElGamal, entonces, los votos cifrados se suelen representar por un vector con tantas posiciones como candidatos u opciones posibles, y en cada posición aparece el elemento correspondiente elevado a 0 o a 1, cifrado con la clave pública de la elección. La razón de cifrar individualmente cada posición es que así se pueden operar individualmente los cifrados de cada posición, correspondientes a un candidato concreto, de varios votos, y obtener el número de veces que ese candidato ha sido seleccionado. Por ejemplo, suponiendo que el voto tiene un máximo de 5 candidatos y el votante ha elegido el segundo y el quinto, el voto se representaría como $(E(g^0), E(g^1), E(g^0), E(g^0), E(g^1))$, donde $E(g^x)$ denota el cifrado del valor g^x usando el algoritmo de ElGamal. La representación de las opciones de voto en un formato conveniente y su cifrado, así como la firma con la clave privada del votante, se hace en el cliente de voto antes de enviarlo al servidor de voto remoto.

Utilizando exponentes para representar las selecciones, si se multiplican los votos cifrados (efectuando dicho producto por separado en cada coordenada del vector que representa el voto), se obtendrá la suma del número

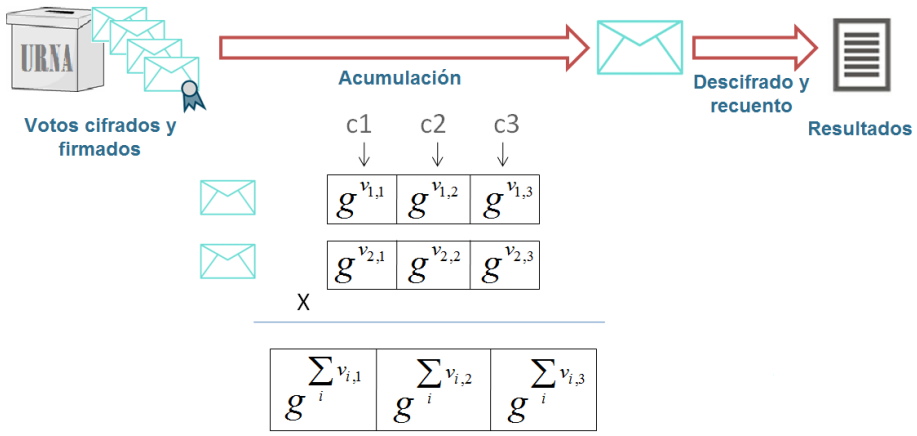

Figura 5: Ejemplo de proceso de recuento homomórfico.

de selecciones que ha recibido cada candidato (ver figura 5). Debe tenerse en cuenta que, después de descifrar el resultado, es necesario calcular el logaritmo discreto en base g para obtener el número de selecciones, o bien se puede tener precalculado g^x para todos los valores x de selecciones posibles.

Los protocolos basados en recuento homomórfico proporcionan una buena protección de la privacidad del votante gracias a las siguientes cualidades:

- No requieren el descifrado individual de los votos, por lo que no existe ningún riesgo de que se puedan relacionar los votos descifrados con los votantes.
- Los votos se cifran antes de ser emitidos (en el terminal del votante).

El problema de este tipo de protocolos está relacionado con la integridad de la elección: al no descifrar los votos uno a uno, en principio no es posible verificar que un votante no ha hecho trampa y ha dado más de un voto a un candidato (por ejemplo poniendo un 2 en lugar de un 1 en el exponente). Para evitar esto, estos protocolos requieren que en el momento de cifrar se generen pruebas criptográficas que permitan verificar que el voto contiene el cifrado del valor 1 o 0. Estas pruebas criptográficas se conocen como pruebas de conocimiento nulo o Zero Knowledge Proofs en inglés [56]. La propiedad de conocimiento nulo hace referencia a que el que hace la prueba puede demostrar a un verificador ciertas propiedades de unos valores que mantiene en secreto (el verificador no llega a conocerlos). En el caso

que nos ocupa, esta propiedad impide que el servidor sepa si la selección del votante ha sido un 1 o un 0, pero puede comprobar que se ha elegido uno de estos dos valores. Estas pruebas criptográficas incrementan de forma sustancial el número de cálculos a realizar en el cliente de voto y están directamente relacionadas con el número de candidatos/opciones de la elección, por lo que son sistemas poco escalables.

2.4. Protocolos de Mezcla o Mixing

Estos protocolos [10] actúan en la fase de recuento. Se basan en imitar las elecciones convencionales, cuando al acabar la elección la urna se agita para romper el orden en que los votos se han depositado.

Los votos emitidos por los votantes, cifrados y firmados previamente a su envío al servidor de votación, se almacenan. Cuando acaba la elección, los votos se separan de sus respectivas firmas digitales (una vez verificadas) y se pasan por un proceso de Mixing para desordenarlos según una permutación secreta. Gracias a este proceso se pierde la relación entre los votos y sus firmas y los votos ya no pueden vincularse con los votantes que los han emitido, con lo que puede procederse a su descifrado para obtener el resultado.

Para realizar este proceso de mezcla se usa una red de nodos denominados Mix-nodes que, por turnos, van recibiendo los votos de salida del nodo anterior y se permutan según unos valores secretos para que el orden original no pueda ser restablecido una vez mezclados (ver figura 6).

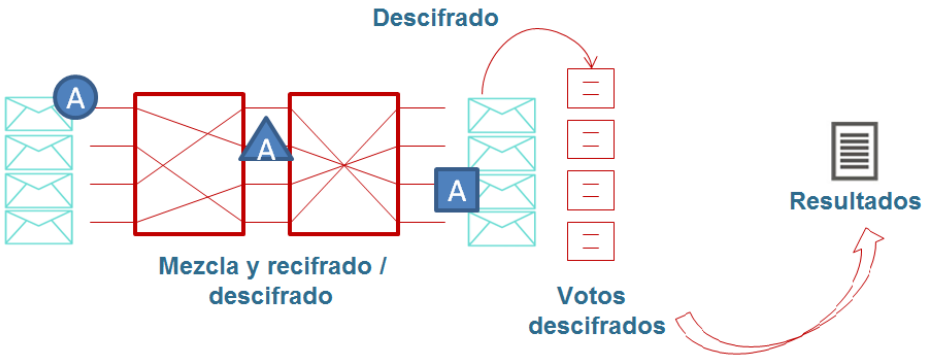

Figura 6: Esquema de recuento basado en una Mixnet.

Para evitar que se pueda restablecer el orden original mediante la comparación de las entradas y salidas de cada nodo, los nodos, además de aplicar

la permutación, aplican una transformación sobre cada voto cifrado que impida su relación con el voto entrante correspondiente. Esta transformación suele consistir en una operación de descifrado o recifrado de forma que se cambia el aspecto de los votos cifrados sin variar su contenido (voto en claro).

2.4.1. Mixnets de descifrado

Presentadas por primera vez en [19], originalmente se usaba el algoritmo RSA para cifrar los votos en este tipo de mixnets. Los votos son cifrados por los votantes en varias capas, tantas veces como nodos hay en la mixnet, y cada vez con la clave pública de un nodo diferente, comenzando por el último y acabando por el primero. Cuando los votos entran en la mixnet, cada nodo permuta los votos y usa su clave secreta para eliminar la capa correspondiente de cifrado de los votos. Este proceso se repite en cada nodo hasta que el último elimina la última capa de cifrado y proporciona los votos descifrados. El principal inconveniente de este sistema es que el votante ha de cifrar su voto tantas veces como nodos hay en la mixnet, y que, debido al relleno (padding) que añade el algoritmo RSA al contenido antes de cifrar, la medida del voto se incrementa con cada capa de cifrado. Con el uso de algoritmos de cifrado como el de ElGamal (puesto en práctica más tarde) estos inconvenientes se eliminan ya que el votante puede cifrar el voto una única vez con la clave resultante del producto de las claves públicas de los nodos de la mixnet y este algoritmo no introduce padding.

2.4.2. Mixnet de recifrado

En este tipo de mixnet se usan algoritmos de cifrado que permiten la realeatorización del voto, sin añadir (hablando en términos prácticos) otra capa de cifrado al voto. Es decir, recifrar un voto cambia su valor (propiedad de realeatorización) y, tras cifrar varias veces el voto sólo hay que descifrar una vez para recuperar su valor en claro. El algoritmo RSA añade un padding aleatorio al valor del voto: $[([[v|\mathrm{pad1}]^e \pmod n)|\mathrm{pad2}]^e \pmod n)|\mathrm{pad3})^e \pmod n$ (donde $|$ denota una operación de concatenación) que hace que no sea posible recuperar el valor del voto descifrando una sola vez, sino que es necesario descifrar tantas veces como se haya cifrado, para sacar en cada paso el padding que se había añadido. ElGamal es un ejemplo de algoritmo con propiedades aleatorizantes que no usa padding. El cifrado de un voto con ElGamal es $E(v) = (g^r \pmod p, vh^r \pmod p)$, donde g, p, h son el generador, el módulo y la clave pública del algoritmo, respectiva-

mente, r es un valor aleatorio y v es el valor del voto. En ElGamal recifrar con la misma clave pública equivale a modificar el valor aleatorio en los exponentes: $E'(v) = (g^r, vh^r) \cdot (g^{r'}, h^{r'}) = (g^{r+r'}, vh^{r+r'})$.

Sólo es necesaria una operación para recuperar v.

Entonces, en este tipo de mixnet cada nodo, en su turno, recifra los votos para evitar que a su salida puedan compararse directamente con los valores a su entrada. Cuando los votos han pasado por todos los nodos de la mixnet, se realiza una única operación de descifrado. Los valores de realeatorización aplicados en cada nodo se mantienen en secreto, igual que la permutación, para evitar que se pueda descubrir el orden original de los votos, que se pueden relacionar con las firmas de los votantes que los han emitido.

En estos protocolos la privacidad del votante se consigue mediante

- El cifrado del voto en el cliente de votación antes de ser emitido.
- La utilización de un proceso de mezcla o mixing para romper la correlación entre los votos cifrados y firmados y los votos descifrados.

El principal inconveniente de los protocolos de mixing es que actúan como una caja negra que transforma y mezcla votos, su seguridad se basa en que los valores de transformación y permutación se mantengan en secreto, por lo que su auditoría, aunque crucial, no es fácil. Si la mixnet no permuta los votos correctamente o revela los valores de la permutación y/o realeatorización aplicados sobre los votos, se podría llegar a correlacionar los votos descifrados con los votantes que los han emitido. Además, dado que la mixnet cambia el valor de los votos que entran para evitar su rastreo, ya sea descifrándolos o recifrándolos, también podría cambiar el contenido de estos votos o sustituirlos sin ser detectado.

Como ventajas de estos protocolos podemos destacar que usan algoritmos de cifrado que permiten una representación del voto más general, permitiendo procesar votos que contengan textos escritos por los votantes (write-ins), y en general pueden ser usados en elecciones complejas.

3. Nuevas variantes

Los protocolos explicados, divididos en cuatro tipos básicos, se han visto ampliados en los últimos años con propuestas que proponen modificaciones para reforzar algunas de las propiedades de seguridad así como para mejorar sus características. A continuación describimos algunas de estas propuestas.

3.1. Mixnets híbridas

En [51, 53] se presenta un esquema de recuento de voto electrónico donde se combinan las técnicas de recuento homomórfico y mixing para sacar provecho de los dos esquemas y mitigar sus inconvenientes. Concretamente, el autor trabaja con las características del recuento homomórfico cuando se aprovechan las propiedades del homomorfismo multiplicativo del algoritmo de cifrado, a diferencia del homomorfismo aditivo, donde el voto para cada opción de voto posible se representa como una constante elevada al valor 1 o 0, dependiendo de si se ha escogido o no la opción de voto, en el homomorfismo multiplicativo cada opción se representa por un número primo y el votante sólo ha de enviar los valores de los números primos correspondientes a las opciones que ha escogido cifrados. De esta forma el voto de un votante que escoge el candidato 1 (representado por un número primo v1) y el voto de un votante que ha escogido el candidato 3 (representado por un número primo v3) pueden multiplicarse para conseguir el cifrado del producto de las dos opciones. Así, una vez descifrado el voto, un proceso de factorización permitirá extraer estos números primos para obtener las opciones elegidas (v1 y v3), sin poder discernir qué opción ha elegido cada votante (ver figura 7).

En este esquema, la privacidad de los votantes se mantiene entre el grupo de votantes el voto de los cuales se opera antes de descifrar el resultado. Un algoritmo con propiedad homomórfica, como ElGamal, requiere (para que se mantenga esta propiedad) que el resultado del producto de las opciones cifradas no sea mayor que el módulo: siendo el cifrado del voto del primer votante $(g^{r_1}, v_1 h^{r_1})$ y el del segundo $(g^{r_2}, v_3 h^{r_2})$, el producto de los dos es $(g^{r_1+r_2}, v_1 v_3 h^{r_1+r_2})$, y el resultado del descifrado es $v_1 v_3 \pmod p$. Si el producto $v_1 v_3$ es mayor que p, la factorización del resultado del descifrado será incorrecta y no dará los valores v_1 y v_3. Podemos entender que un número limitado de votos pueden ser operados antes de descifrar en caso de que los números primos asignados a la elección sean relativamente grandes (por ejemplo en una elección a gran escala donde hay muchas

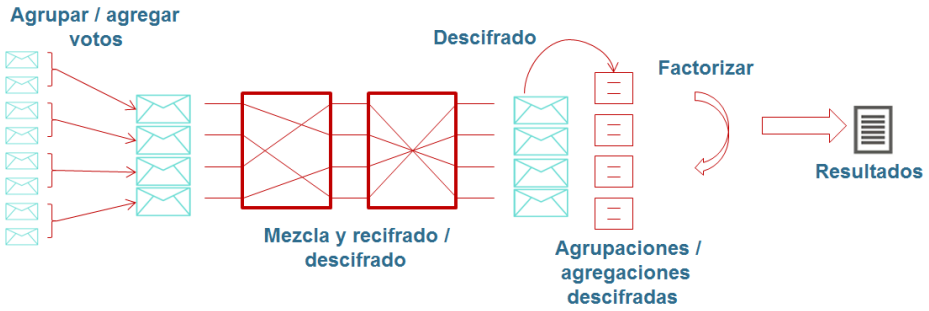

Figura 7: Mixnet híbrida.

opciones). Para prevenir problemas de privacidad en el caso que el subconjunto de votos entre los que esté el voto de un votante específico sea demasiado pequeño para mantener un nivel adecuado de anonimato del votante dentro del subconjunto de votantes que han emitido esos votos, la propuesta contempla la posibilidad de introducir los resultados de las operaciones de cada grupo de votos en una mixnet antes de descifrar, de forma que no pueda relacionarse el resultado de un descifrado con un grupo de votantes específico.

Igual que en los protocolos clásicos de recuento homomórfico, como se operan varios votos antes de descifrar, es importante que cada votante envíe, con su voto cifrado, una prueba criptográfica de conocimiento nulo que demuestre que el voto cifrado que envía contiene una de las opciones válidas de la elección, sin mostrar cuál, dado que si esto se descubre al descifrar, no se podrá saber cuál de los votos operados antes de descifrar era inválido.

Como se ha explicado antes, estas pruebas son especialmente costosas. Por ello, en [52] se propone un esquema para agrupar estas pruebas, de forma que el esquema de mixnet híbrido combinando homomorfismo multiplicativo, mixnets y pruebas criptográficas más eficientes aprovecha las ventajas del recuento homomórfico y las mixnets, mientras que reduce sus inconvenientes.

3.2. Vector-Ballot

Vector-Ballot [42] es una propuesta que pretende hacer compatibles los esquemas de recuento homomórfico con la posibilidad de que los votantes introduzcan respuestas escritas por ellos mismos (write-ins). La idea principal es construir un voto igual que en el recuento homomórfico aditivo

(aunque también podría utilizarse en un esquema homomórfico multiplicativo), es decir, una constante elevada al valor 1 o 0 indica para cada posible opción de la elección si el votante la ha elegido o no. En caso de tener la posibilidad de llenar algún campo con texto libre, esta opción también se representa como el cifrado de la constante elevado al valor 1 o 0 dependiendo de si esta opción se ha llenado o no, actuando como un tipo de indicador. En caso de que se haya llenado, el texto introducido por el votante va cifrado y se coloca a continuación de las otras opciones de voto (ver figura 8).

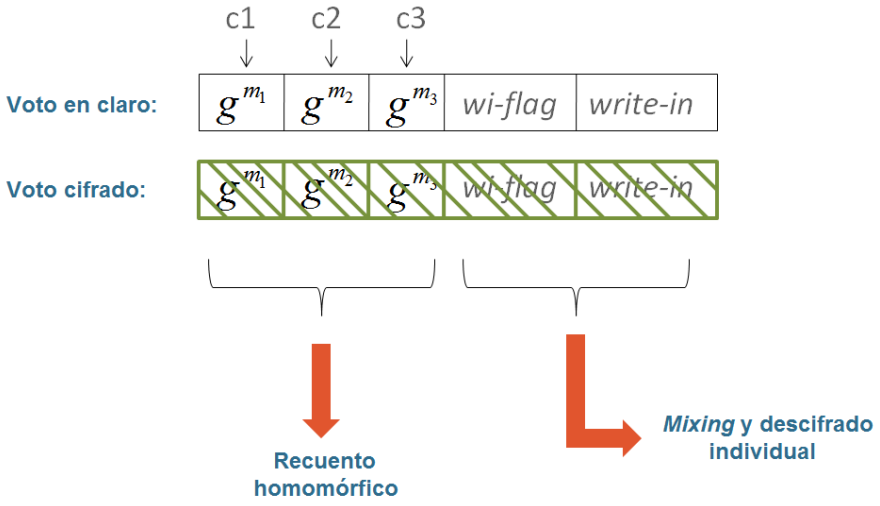

Figura 8: Estructura del voto en Vector-Ballot.

Para impedir que un observador externo pueda inferir qué votantes han llenado esta opción, el cifrado del texto libre se incluye en todos los votos, aunque para los votantes que no han puesto nada este texto está vacío (al utilizar un cifrado con propiedades aleatorizadoras un observador externo no lo podría distinguir). El valor cifrado del indicador, que proporciona información sobre si hay contenido en el texto libre cifrado o no, se usa para saber cuáles son los votos que contienen texto libre introducido por el votante. En el momento del recuento, cada voto se divide según los contenidos que contiene: el campo de texto libre cifrado de los votos, donde los votantes han introducido un contenido, se introducirá en una mixnet para desvincularlo del votante que lo ha emitido, antes de descifrarlo. En cambio, la parte del voto correspondiente a las opciones que han sido seleccionadas entre un conjunto predeterminado, se pasa por un proceso de

recuento homomórfico aditivo para extraer los resultados. Así, la propuesta adapta los esquemas de recuento homomórfico para su uso en elecciones en las que haya la opción de respuesta libre.

3.3. Mixing de credenciales

El artículo Coercion-Resistant Electronic Elections [39] presenta una propuesta de esquema de voto electrónico enfocada a evitar la venta de votos o coacción de los votantes. En protocolos como los de recuento homomórfico o de Mixing, el servidor de voto recibe votos cifrados y firmados, que se pueden relacionar directamente con los votantes que los han emitido.

Los autores del artículo hacen énfasis en el hecho de que un coaccionador podría entregar al votante el voto cifrado con la opción deseada, y comprobar que éste se encuentra almacenado en el servidor de voto, firmado digitalmente por el votante: como veremos más adelante, es común que los sistemas de voto electrónico hagan pública la lista de votos que recibe el servidor de voto para facilitar la verificación del proceso. Un coaccionador podría incluso asegurar que alguien se abstiene en la votación comprobando que no hay ningún voto recibido por el servidor de voto con su firma digital. La idea principal de este esquema es mantener en secreto la relación entre una firma y la identidad del votante que la ha hecho. Así, el coaccionador no puede localizar, en la lista de votos publicados, el voto emitido por un votante específico.

Normalmente denominamos credenciales a los datos que utiliza un votante para demostrar que puede participar en una elección. En este esquema, la credencial de un votante es su pareja de claves de firma, ya que el sistema aceptará los votos en base a si están firmados con claves válidas.

El esquema (ver figura 9) asume que existe una manera de entregar estas credenciales a los votantes, de forma privada en un momento previo a la fase de voto, de manera que sólo el votante conoce qué credencial le ha sido asignada. Un servicio de registro mantiene un listado con los nombres de los votantes autorizados a participar en la elección y sus credenciales, cifradas para mantener el secreto. El listado contiene en cada entrada el nombre del votante i-ésimo V_i y la credencial asignada (σ_i) cifrada $S_i = E(\sigma_i)$. En la fase de voto, el votante i-ésimo envía el voto v y su credencial cifrados: $A = E(\sigma_i), B = E(v)$. En caso que esté siendo coaccionado, el votante puede emitir un voto falso en el que el voto cifrado es el que ha elegido el coaccionador, y la credencial cifrada es falsa: $A = E(\alpha), B = E(v)$. También puede entregar al coaccionador la credencial falsa para que emita el voto en su nombre. El servidor de voto publica todos los votos que

recibe, así que el coaccionador puede ver el voto publicado aun no sabiendo si la credencial adjunta es válida.

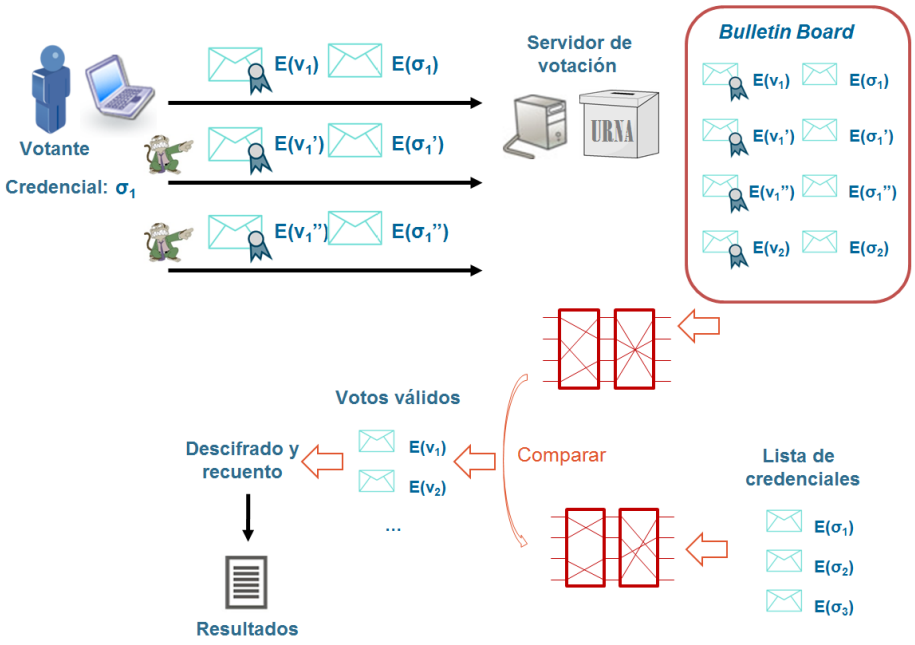

Figura 9: Flujo del protocolo de Mixing de credenciales.

Acabada la fase de votación, la lista de votos recibidos, compuesta por parejas de cifrados de opción de voto y credenciales (A, B), y la lista de credenciales cifradas mantenida por el servicio de registro (S) se mezclan utilizando una mixnet de recifrado, de forma que las entradas de los votos en la lista mezclada no pueden relacionarse con la lista de los votos publicada, y la relación entre credenciales cifradas y los votantes a los que se habían asignado, V_i, S_i, se pierde. Finalmente, mediante pruebas criptográficas de conocimiento nulo, que permiten comprobar que dos valores son el cifrado del mismo texto en claro [37], se puede comprobar qué votos han sido emitidos con credenciales válidas, es decir, qué parejas (A, B) contienen en A el cifrado de una de las credenciales de la lista de valores S. Estos votos constituyen la lista de votos válidos que se descifran y se cuentan para obtener los resultados de la elección. Como antes de la validación de los votos, éstos han sido procesados por la mixnet, un coaccionador no podrá saber si el voto elegido por él está incluido en la lista de votos válidos o no.

Las pruebas que permiten comprobar la coincidencia entre los valores A y S_i son muy costosas. En caso de que haya un gran número de votos porque

han participado muchos votantes o porque se han emitido muchos votos con credenciales falsas, la realización de todas las pruebas (cuyo número tiene una relación cuadrática con el número de votos emitidos) puede impedir que los resultados de la elección se obtengan en un tiempo razonable.

Por esta razón el esquema es vulnerable a ataques de denegación de servicio (mediante el envío de muchos votos con credenciales falsas) y poco adecuado para elecciones a gran escala.

Civitas [22] es una implementación práctica de este esquema. [63] es una mejora del esquema original de [39] donde se aumenta la eficiencia de la detección de votos emitidos con credenciales falsas. La idea principal es que el votante indica cuál es la entrada de la lista de credenciales mantenida por el servicio de registro (las parejas V_i, S_i) correspondiente a su identidad (V_i), enviando este valor cifrado juntamente con el voto, de forma que sólo las entidades encargadas del recuento podrán descifrarlo. Así se sabe qué parejas de valores (A, B) han sido emitidas por un votante específico y las pruebas criptográficas para comprobar qué parejas han sido emitidas con credenciales falsas, solo se tendrán que hacer comparándolas con el valor S_i asignado a este votante, y no con toda la lista de valores S como pasa con el protocolo original. Dado que las credenciales están cifradas, y que los valores (A, B) se pasan por una mixnet antes de su validación, los autores reivindican que el nuevo esquema mantiene las mismas propiedades de seguridad que el original. En [43] se presenta otra mejora de la propuesta [39] donde se propone limitar el número de votos con credenciales inválidas que puede emitir un votante, para así asegurar que la lista de votos recibidos que han de ser validados mediante pruebas criptográficas no crece de forma desmesurada. La idea básica consiste en entregar a cada votante una credencial válida y una lista limitada de credenciales inválidas que puede usar para engañar al coaccionador, de forma que el coaccionador no puede distinguir cuáles son válidas y cuáles no, pero el proceso de validación de los votos (que se realiza acabada la fase de voto) sí que puede distinguirlas para contabilizar solo los votos emitidos con credenciales válidas. Durante la fase de voto, el servicio de voto comprobará que los votos que recibe contienen una credencial cifrada de las de una lista donde se encuentran las válidas y las inválidas entregadas a todos los votantes.

4. Verificabilidad

4.1. Problemática: no son personas, es software

Los protocolos anteriormente presentados funcionan en el cumplimiento de los requisitos de seguridad, en cierta medida y dentro de las capacidades que se han mencionado en la descripción, siempre que confiemos en que los agentes que ejecutan el protocolo son honestos. Ésta, por ejemplo, es una de las limitaciones del voto electrónico en general, y también del voto por internet: en el momento de emitir el voto, el que ejecuta el protocolo no es el votante, sino el software de voto. ¿Cómo podemos saber que hace lo que el votante quiere?

Lo mismo sucede con el proceso de recuento. Como los votos son digitales, no los cuentan los miembros de la mesa electoral, como en una elección tradicional, sino un software de recuento. En el caso de utilizar técnicas de anonimización en la fase de recuento, como el recuento homomórfico o el mixing, el software no solo está a cargo de contar votos, sino de hacer un procesado previo (ya sea la operación de los votos cifrados o el mixing) y descifrarlos para poder contar. Entonces, es importante poseer alguna herramienta que permita auditar este código para garantizar que durante todas estas operaciones no se modifiquen los votos o el resultado obtenido del recuento.

Algunas medidas que se pueden aplicar son: realizar una auditoría del código para asegurar que el funcionamiento es el esperado, hacer una compilación pública (lo que se conoce como trusted build) del código auditado, y garantizar que este mismo código es el que se ejecuta, usando código firmado digitalmente, boot CDs para asegurar que el código se arranca en un sistema limpio (sin virus o códigos maliciosos), o herramientas de auditoría del software en ejecución. De estas últimas, las herramientas que escanean los ficheros ejecutables y de configuración para detectar cualquier modificación son muy populares (Tripwire, AIDE), pero presentan la siguiente cuestión: ¿Quién fue primero, el huevo o la gallina? Si utilizo una herramienta para escanear los ficheros ejecutables del sistema y comparar que coinciden con los auditados, quién asegura que esta herramienta no ha sido modificada, es más, estas herramientas pueden escanear los ficheros ejecutables, pero no ven cuáles son los que efectivamente se acaban ejecutando. ¿De qué nos sirve comprobar que el software bueno está instalado si después se ejecuta otro? Una mejora de estos sistemas de escaneo es el Integrity Measure Architecture (IMA), una herramienta presente a partir de la versión 6 de las distribuciones RedHat y CentOS de Linux, que genera un registro

(un fingerprint) de cada fichero antes de que se ejecute. Así se puede conocer el código ejecutado y no solo el instalado. Esta herramienta puede usarse conjuntamente con un TPM (Trusted Platform Module [5]), un chip hardware criptográfico propuesto por el Trusted Computing Group [6] y presente hoy en día en muchos ordenadores, que, entre otras cosas, almacena de forma segura registros o fingerprints de los valores de los procesos ejecutados en un ordenador, desde que se pone en marcha la BIOS hasta que se arranca el sistema operativo, lo que permite mantener la cadena de confianza hasta el inicio de la herramienta IMA. La IMA además se puede configurar para que almacene los fingerprints que genera en los registros seguros del TPM, para evitar modificaciones o borrados.

4.2. Herramientas criptográficas de verificación

Como hemos visto, auditar la autenticidad del software que se ejecuta no es trivial. Y aún menos cuando el software se ejecuta en ordenadores remotos no controlados, como pueden ser los de los votantes. El cliente de votación puede verse afectado por virus o malware que esté infectando el dispositivo del votante, de forma que cambie las opciones escogidas por éste y envíe al servidor de voto un voto que contiene otras. El votante, incluso pudiendo ver la comunicación entre su dispositivo y el servidor, no podría saber si lo que se está enviando coincide con lo que ha elegido, ya que el voto va cifrado.

Por ello, una evolución que se ha experimentado recientemente en el voto por Internet, es la demanda de verificación, que consiste en que los votantes, auditores y otros observadores externos, sean capaces de poder verificar que el software de voto funciona de forma correcta. Así se puede asegurar que los resultados de la elección son correctos y representan la intención de voto de los votantes, sin poner en riesgo la privacidad.

Hemos visto que hay dos problemáticas básicas: que el votante pueda asegurar que el voto que emite representa sus opciones, y que se pueda auditar el proceso de anonimización de los votos y recuento. De los requisitos de seguridad del voto electrónico podemos extraer otras, como poder verificar que sólo se cuenta un voto por votante, y que estos votos han sido emitidos por votantes válidos. Existen varias clasificaciones según el tipo de verificabilidad (ver figura 10):

- En [33] se habla de clasificación entre verificabilidad individual y verificabilidad universal:

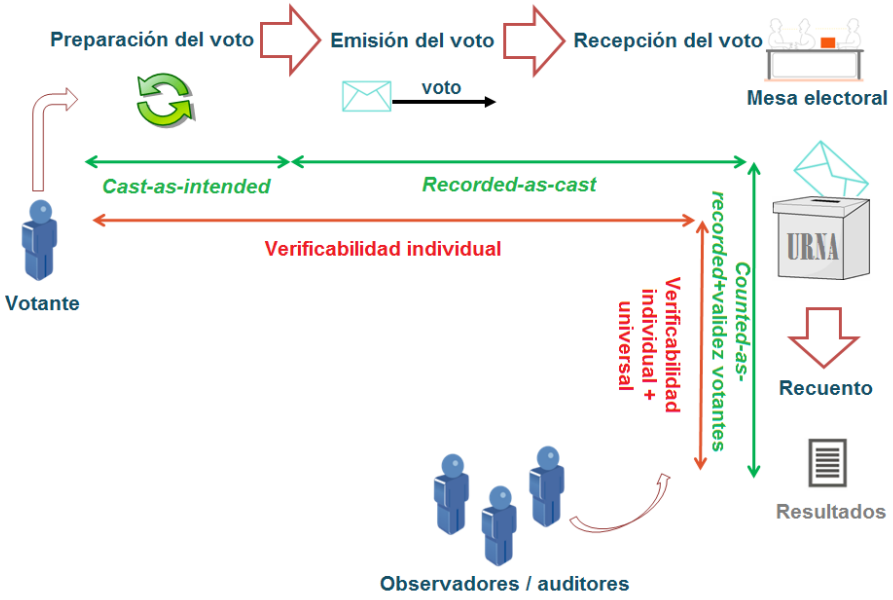

Figura 10: Conceptos de verificabilidad.

1. La verificabilidad individual se refiere normalmente a la posibilidad de que los votantes puedan verificar que sus votos se incluyen en el recuento final [36].
2. La verificabilidad universal se refiere a la posibilidad de que cualquier observador pueda verificar que el recuento se ha hecho correctamente. Algunos autores incluyen la propiedad de que se pueda verificar que los votos han sido emitidos por votantes legítimos.

- Por otro lado existen también las nociones de cast-as-intended y counted-as-cast :

  1. La verificabilidad cast-as-intended consiste en poder verificar que el voto emitido representa las opciones escogidas.
  2. La verificabilidad counted-as-cast consiste en poder verificar que el proceso de recuento ha sido correcto.

- En [15] se habla de la noción de end-to-end verifiability, o verificación punto-a-punto: los votantes y observadores externos (o audito-

res) han de poder verificar que todos los votos han sido emitidos por votantes válidos, y que se cuentan correctamente.

- Aún se puede refinar más y hablar de end-to-end verifiability como una propiedad que se puede dividir en tres: cast-as-intended verifiability, recorded-as-cast verifiability y counted-as-recorded verifiability, donde se incluyen los procesos de verificación de que los votos emitidos se corresponden con las opciones escogidas por los votantes, que los votos se guardan tal y como se reciben y no son modificados hasta el momento del recuento, y que los votos se cuentan correctamente.

Una característica muy buena de la criptografía es que está basada en matemáticas y las matemáticas no mienten. Quizás no pueda asegurarse que un software específico es auténtico o se ejecuta sin interferencia externa, pero podemos forzar a este software a hacer pruebas criptográficas que demuestren que las operaciones que hace las hace correctamente, sin que pueda mentir. A continuación explicaremos algunos protocolos criptográficos de votación y herramientas que aportan verificabilidad al proceso.

4.3. Verificación cast-as-intended

Existen principalmente dos tipos de métodos para realizar esta verificación: desafiar al cliente de votación o utilizar códigos de retorno.

4.3.1. Desafiar al cliente de votación

Como hemos introducido en la sección 1.5, el cliente de Votación consiste en el software que se ejecuta en el dispositivo del votante. Este cliente normalmente se encarga de presentar al votante las opciones de voto, recoger sus selecciones y cifrar y firmar el voto antes de enviarlo al servidor remoto de votación.

El proceso de verificación consiste en seguir los pasos de la aplicación de voto hasta que las opciones de voto elegidas son cifradas por el cliente de votación, y después tener la oportunidad de pedir al cliente que abra el voto cifrado para ver que lo que había dentro y se iba a enviar era efectivamente lo que se había elegido (ver figura 11). Este método fue propuesto en principio por Josh Benaloh [15] y se utiliza en el sistema de voto por Internet Helios [11, 12], un sistema que empezó como una prueba académica y que se ha utilizado en muchas elecciones universitarias y cada vez gana más atención. La idea es que una vez ha cifrado el voto, el cliente de votación

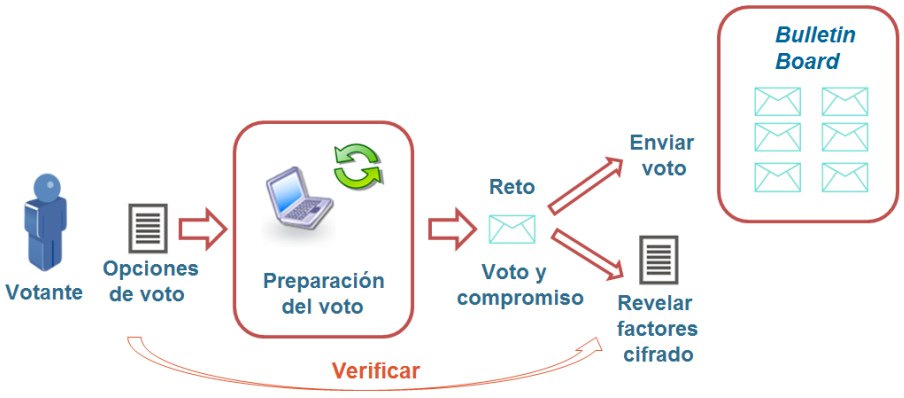

Figura 11: Verificación cast-as-intended con reto al cliente de voto.

se compromete mostrando al votante un hash del valor del voto cifrado. Con comprometerse, queremos decir que el cliente de votación da un dato referente al voto cifrado (en este caso, el valor de su hash) de forma que más tarde no puede retractarse y decir que el valor del voto cifrado era otro. Antes de enviar el voto, el votante tiene la oportunidad de auditarlo y pedir al cliente de voto que muestre los valores con los que ha cifrado el voto (por ejemplo, en caso del algoritmo de cifrado de ElGamal mostraría el valor aleatorio que ha usado para cifrar), para que el votante pueda comprobar que cifrando las opciones escogidas con los valores entregados por el cliente, el hash del resultado coincide con el compromiso dado. En caso de auditar el voto, éste se vuelve a cifrar con otros valores diferentes antes de ser enviado para que el votante no pueda usar los valores entregados por el método de verificación, para demostrar a un tercero cómo ha votado. El servidor de voto remoto dispone de una herramienta de publicación de los votos recibidos, normalmente conocida como bulletin board. Este bulletin board juega un papel muy importante en los protocolos de voto verificables. En [44] se explica que un bulletin board es un canal público donde los participantes autorizados pueden publicar datos, y, una vez publicados, no pueden ser ni borrados ni sobrescritos por nadie. Entonces, el votante puede comprobar que, en caso de decidir enviarlo, el voto cifrado al que el cliente de voto se había comprometido, antes de que el votante eligiera entre enviarlo o auditarlo, es el que ha recibido el servidor de voto. Como el cliente de voto se compromete al valor del voto cifrado antes de saber si el votante lo auditará o no, no puede comprometerse a un voto diferente al que enviará sin que haya un 50 % de probabilidad de que el votante le

descubra.

El problema de este sistema es que el votante necesita una herramienta para rehacer el cifrado de las opciones escogidas con los valores entregados por el cliente de voto retado, para comprobar que coincide con el valor comprometido. Además del problema de que los votantes entiendan el sentido de los números largos que se les muestra, claves públicas, aleatoriedad, etc., hay que entregar esta herramienta (que no podría ser una calculadora dada la longitud de los números para hacer los cálculos) de forma independiente al software del cliente de voto, para que no se pongan los dos de acuerdo para engañar al votante.

Otras propuestas de voto electrónico que implementa este tipo de verificación son VoteBox [60] y Wombat Voting [9].

VoteBox es un sistema de voto electrónico presencial, pensado para elecciones donde se vota utilizando máquinas de voto electrónico (también conocidas como Digital Recording Electronic voting machine, o DRE) en los colegios electorales.

El voto electrónico presencial es muy común en Estados Unidos. En este sistema, todos los DREs de un colegio electoral están conectados mediante una LAN (Local Area Network). Cuando el votante emite el voto, el voto cifrado se almacena en el DRE donde está el votante y se transmite a través de la LAN a los otros DREs, que también la almacenan. Una vez emitido el voto, si el votante quiere auditarlo, el DRE envía a través de la LAN los factores que ha utilizado en el cifrado. En el mismo colegio hay montado un dispositivo llamado diodo que pincha la LAN, de forma que se puede observar todo el tráfico de la red. Este diodo se utiliza para observar, tanto el voto cifrado que se ha enviado a todos los DREs, como los factores de cifrado que entrega el DRE en cuestión si el votante decide auditar su voto. Con los datos observados se puede verificar que el voto emitido se corresponde con el elegido por el votante. Una vez auditado, el voto almacenado en todos los DREs se anula para prevenir la venta de votos (similar al caso de Helios).

Wombat Voting es un sistema de voto presencial donde los votantes utilizan máquinas de votación para seleccionar sus opciones e imprimir sus votos en papel para depositarlos en una urna electoral. Estos votos (cifrados) son impresos en un formato de código de barras bidimensional (o QR), de forma que se puede utilizar un lector para digitalizar los votos y hacer el recuento de forma electrónica. De forma muy similar al sistema Helios, una vez el voto cifrado se imprime en papel, el votante puede decidir si auditar el voto, con lo que la máquina de votación entrega los valores del cifrado para comprobar que lo que había impreso es el cifrado de las opciones

escogidas por el votante. Igual que en los otros sistemas, los votos auditados no pueden ir a la urna.

4.3.2. Los códigos de retorno

Son una de las funcionalidades de los protocolos Pollsterless de los que hablamos en la sección 2. La principal idea es que el servidor de voto remoto utiliza una clave secreta para generar nuevos valores a partir del voto cifrado que ha enviado el votante. Estos valores o códigos de retorno se envían al votante, que está en posesión de una tarjeta especial donde aparecen todas las opciones de voto posibles con los correspondientes códigos de retorno asignados (ver figura 12). Si los códigos recibidos se corresponden con los que en la tarjeta están asignados a las opciones escogidas, suponiendo que los códigos de retorno que hay en las tarjetas sólo son conocidos por los votantes, y que la clave del servidor de voto se mantiene secreta, el votante puede estar seguro de que las opciones cifradas recibidas en el servidor son las mismas que las que él ha escogido.

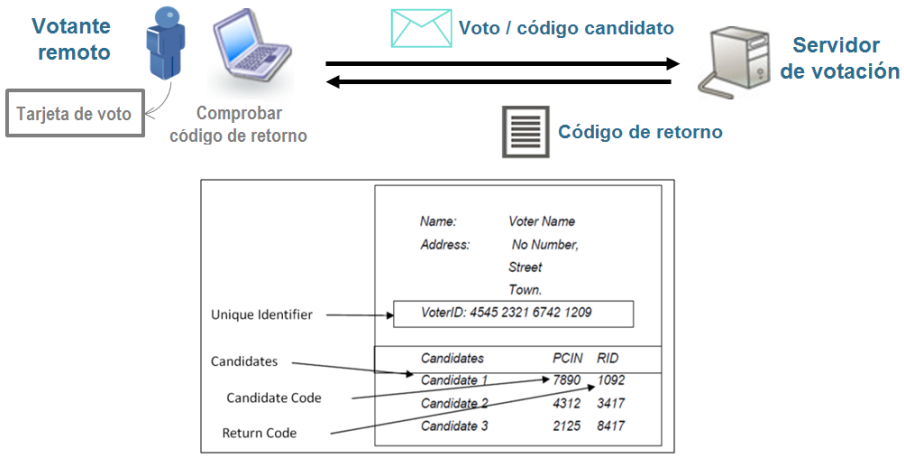

Figura 12: Verificación cast-as-intended con códigos de retorno.

Como hemos visto en la sección anterior, los esquemas de Pollsterless utilizan dos tipos de códigos, uno de candidato u opción, que es el valor precifrado de la opción escogida que se envía al servidor de voto, y el código de retorno que envía el servidor como respuesta al votante. También existe la opción mixta de tener un cliente de votación que sea el que cifre las opciones escogidas (sin usar valores precifrados) y obtener unos códigos de retorno generados por el servidor como confirmación de estas opciones.

En este caso, las tarjetas de los votantes solo tendrían códigos de retorno, y no códigos de candidato u opción. Esta variante del Pollsterless presenta algunas ventajas: los votantes no han de poner códigos para votar, sino compararlos para verificar, lo que comporta una gran mejora de la usabilidad del sistema. Además, estamos hablando de utilizar tarjetas en papel que han de ser distribuidas. Como ya se ha dicho, una de las ventajas del voto electrónico es poderse ahorrar toda la logística asociada al reparto de papeletas, voto postal etc. En un sistema Pollsterless puro, en caso de que las tarjetas no lleguen a tiempo a los votantes, estos no podrán votar. En caso de la opción mixta, los votantes podrán votar, pero no verificar, lo que puede considerarse un mal menor. Incluso podría ser que los votantes tuvieran las tarjetas antes de que acabe el periodo de votación, para poder verificar el voto que habían emitido en su momento y estar a tiempo de anularlo si el código de retorno no coincide con la opción escogida.

Para poder aplicar la funcionalidad de los códigos de retorno, es necesario que las opciones de voto se cifren con un algoritmo con propiedades deterministas, de manera que cuando el servidor de voto haga sus cálculos sobre estas opciones cifradas, usando su clave secreta, genere unos códigos de retorno que hayan podido ser calculados previamente a la fase de configuración de la elección, para generar las tarjetas de los votantes.

Obviamente, con un sistema de cifrado con propiedades aleatorizadoras, los valores no serían repetibles. Por otro lado, desearíamos que los votos emitidos al servidor de voto se hubieran cifrado con un algoritmo con propiedades aleatorizadoras para no poder saber si varios votantes han votado por la misma opción, o si un votante envía dos votos con el mismo contenido en caso de que se permita el voto múltiple (también para evitar coacción o venta de votos). Las opciones que se han propuesto hasta ahora pasan por tener un servidor de voto que sea capaz de quitar la capa de cifrado aleatorizador para generar los códigos de retorno [34], o para generar dos tipos de cifrado en el cliente [55], uno que puede destapar el servidor para tener unos valores deterministas sobre los que generar los códigos de retorno, y otro que no, y que es el que se hará llegar al proceso de recuento.

En el primer caso, el servidor de voto está dividido en dos componentes que, juntas, tienen la capacidad de descifrar los votos (podemos decir que cada uno tiene la mitad de la clave privada de la elección). Para proteger la privacidad de los votantes, la primera componente sacará la parte del cifrado aleatorizador correspondiente a su clave privada, pero añadirá una capa de cifrado determinista antes de pasar el resultado a la segunda componente, que sacará la segunda parte del cifrado aleatorizador. El principal inconveniente de esta solución es que si las dos componentes se ponen de

acuerdo, pueden descifrar todos los votos y romper la privacidad de los votantes.

En el segundo caso, el cliente de votación hace dos cifrados de las mismas opciones: uno con un algoritmo con propiedades aleatorizadoras, usando la clave pública de la elección, y otro que en un primer paso se cifra con un algoritmo determinista, y después con un algoritmo con propiedades aleatorizadoras con la clave pública del servidor de votación. Así, el servidor de votación es capaz de sacar la capa aleatorizadora del segundo cifrado para obtener un valor determinista sobre el que generar el código de retorno, pero no puede ver cuál es la opción escogida por el votante (porque no puede sacar la capa del cifrado determinista). Se utilizan pruebas criptográficas para demostrar que los cifrados sobre los que se generan los códigos de retorno contienen las mismas opciones de voto que los otros cifrados, que son los que se procesan en la fase de recuento. Estas pruebas criptográficas son pruebas de conocimiento nulo que, igual que en el sistema de Mixing de credenciales de la sección 3, se utilizan para demostrar que dos cifrados se corresponden con el mismo texto en claro sin tener que mostrar los valores secretos (el valor o la opción de voto cifrada).

Los principales inconvenientes de esta solución son la usabilidad y el alto coste computacional que recae en el cliente de voto: el cliente de voto ha de cifrar cada opción de voto por separado, para poder generar un código de retorno independiente para cada una; como hemos explicado, no se hace un único cifrado, sino dos para cada opción de voto (uno sobre el que se genera el código de retorno y otro que va al recuento), lo que conlleva un gran número de operaciones por selección; la generación de pruebas criptográficas que relacionan un cifrado con el otro (para asegurar que tienen el mismo contenido) añade más operaciones al cliente de voto; finalmente, el cifrado con propiedades deterministas que hace el cliente de voto ha de utilizar algún secreto que solo conozca el votante, para evitar que el servidor descubra demasiada información al generar los códigos de retorno. La solución propuesta es que el votante introduzca un código único que viene con la tarjeta de los códigos de retorno, que se asuma que solo él la conoce. Como se ha explicado, esto va en contra del requisito de que el votante no haya de introducir códigos para verificar su voto.

Aunque mucha gente trabaja en el mismo problema, no se ha encontrado una solución óptima, sino que se elige la que tiene menos inconvenientes según el criterio que se establezca. Esto demuestra que aún hay mucho camino por recorrer.

4.4. Verificación counted-as-cast

Los sistemas de verificación que actúan en esta fase se aplican habitualmente en esquemas de voto electrónico donde se utilizan técnicas de recuento homomórfico o Mixnets, ya que son estos procesos que anonimizan los votos los que hay que auditar para asegurar que los resultados de la elección se corresponden con lo que han votado los votantes.

4.4.1. Esquemas de recuento homomórfico

Se basan en la propiedad homomórfica del algoritmo utilizado para cifrar los votos, que hace que la operación de votos cifrados tenga como resultado el cifrado de la operación de los votos. Se usa esta propiedad para obtener el resultado de la elección de forma verificable de la siguiente manera (ver figura 13):

Figura 13: Recuento homomórfico verificable.

- Los votos emitidos por los votantes, cifrados y firmados, se publican en un bulletin board al que tiene acceso el servidor de voto remoto. Como los votos cifrados están digitalmente firmados, cualquiera puede comprobar que los votos han sido emitidos por votantes válidos.
- Finalizada la fase de voto, los votos cifrados se operan entre ellos, y el resultado también se publica en el bulletin board, de forma que cualquiera puede repetir la operación y ver que el resultado es correcto.
- La clave privada de la elección se utiliza para descifrar el resultado de operar los votos. Como ya hemos explicado en la sección 2, dependiendo del tipo de homomorfismo, resultará el producto de las opciones de voto escogidas por los votantes, o la suma. En el caso de la suma tenemos el resultado de la elección de forma directa, mientras que en el caso del producto se tendrán que extraer las opciones

una a una para poderlas contar. En este segundo caso, se suelen usar números primos pequeños para facilitar el proceso.

- Para que cualquiera pueda comprobar que el valor descifrado corresponde con lo que estaba cifrado, se publican en el bulletin board pruebas criptográficas de conocimiento nulo que permiten demostrar que el descifrado se ha hecho de forma correcta, sin mostrar el valor de la clave privada. De esta manera, cualquiera las puede validar siguiendo con el principio de verificabilidad universal.

Una medida adicional en estos esquemas es que la clave privada de la elección no exista en ningún momento, ya que se podría utilizar para descifrar los votos individuales en lugar de los acumulados, rompiendo así la privacidad de los votantes. En lugar de esto, se usan medidas como las explicadas en la sección 1.6 para dividir la clave en trozos o shares que están repartidos entre los miembros de la mesa electoral siguiendo un esquema umbral, o incluso, generar los shares directamente sin que la clave privada haya existido nunca. En [23] se define un sistema de voto electrónico con recuento homomórfico, en el que cada miembro de la mesa electoral usa su parte de la clave privada para hacer un descifrado parcial del resultado de la operación de los votos, y genera una prueba de que este descifrado parcial es consistente con su parte de la clave. Esta información se publica en el bulletin board, así cuando el conjunto mínimo de la mesa electoral (según el umbral elegido) haya publicado sus operaciones, los valores de los diferentes descifrados parciales se podrán combinar de forma pública para obtener el resultado.

Los esquemas de recuento homomórfico son muy adecuados para proporcionar propiedades de verificabilidad, ya que casi todas las operaciones se pueden hacer de forma pública y, para las que no (descifrados parciales, etc.), se pueden proporcionar pruebas universalmente verificables que demuestren que se han hecho bien. Por universalmente verificables entendemos que, al no requerir de datos secretos para hacer su validación, pueden ser verificadas por cualquiera. No obstante, los inconvenientes de estos esquemas, comentados en la sección 2, hacen que sean poco flexibles y costosos en general.

4.4.2. Mixnets

Como hemos explicado en la sección 2, son procesos en que varios nodos colaboran para, de forma secuencial, mezclar y recifrar o descifrar los votos con el objetivo de eliminar la relación entre el votante que ha emitido el

voto y el voto en claro. La Mixnet ha de actuar como una caja negra y la permutación y los valores de recifrado o descifrado han de mantenerse en secreto para evitar romper la privacidad del votante, ya que el propósito de la Mixnet es que no se puedan conectar los votos a la salida con los votos a la entrada (que se pueden relacionar con los votos firmados emitidos por los votantes), pero esto también tiene el inconveniente de que no podemos asegurar la correspondencia entre los votos de entrada y de salida a efectos de conocer que la mixnet no ha modificado, añadido o eliminado votos. La solución clásica pasa por ejecutar la mixnet en un entorno aislado y con un código auditado [57]. No obstante, el propósito de la verificabilidad es no tener que fiarnos del software ni de la plataforma, sino que el proceso proporcione unas pruebas de que ha actuado de forma honesta, que no se puedan falsificar y que todo el mundo pueda validar. Desde la Mixnet verificable de Sako y Kilian [59] han surgido muchas propuestas para la generación en cada nodo de la mixnet de pruebas que demuestren que el conjunto de votos a la salida es el recifrado o el descifrado de los de entrada permutados (ver figura 14). Un buen compendio de estas propuestas se encuentra en [10], y básicamente se pueden dividir entre propuestas de verificación heurística y propuestas de verificación exhaustiva.

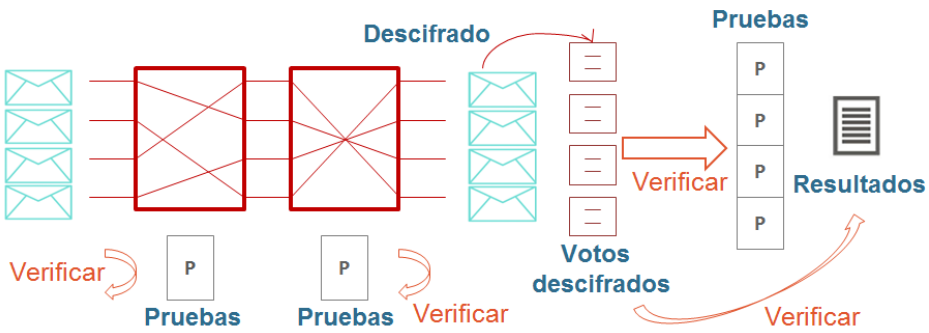

Figura 14: Generación y comprobación de pruebas criptográficas en Mixnets verificables.

En las propuestas de **verificación heurística** los nodos de la mixnet generan pruebas criptográficas que aseguran, con una cierta probabilidad, que la mixnet no está actuando de forma fraudulenta. Estas propuestas se basan generalmente en hacer pruebas de conocimiento nulo que demuestran que un voto a la salida del nodo es el recifrado o el descifrado de un voto a la entrada de este nodo. Por la naturaleza de estas pruebas, no es necesario conocer datos secretos de la mixnet (permutación y factores de recifrado

o descifrado), así que el proceso puede considerarse universalmente verificable. Como mostrar esta relación entre entrada y salida rompe con el propósito de la mixnet de no proporcionar rastreabilidad, existen varias propuestas para generar las pruebas de diferentes maneras, de las cuales explicamos las más significativas:

- El sistema de Random Partial Checking [38] y su mejora [21] proponen que cada nodo descubra las relaciones entre parejas de voto entrante/voto saliente para hacer estas pruebas de recifrado o descifrado, pero solo con la mitad de los votos del nodo. Esta mitad se escoge de forma aleatoria en el primer nodo una vez ya se ha hecho el mixing, para que el nodo no pueda adivinar en el momento del mixing qué votos serán auditados y, por tanto, cuáles puede modificar sin ser detectado. Los otros nodos revelarán en su turno las relaciones entre voto entrante/voto saliente sobre la mitad de sus votos, de manera que el camino de un voto no pueda ser seguido por toda la mixnet: por ejemplo, en una mixnet de dos nodos, el segundo nodo revelaría las relaciones entre parejas de voto entrante/voto saliente de los votos cuya relación no ha sido revelada por el primer nodo.

- En [35] se propone un método en que se multiplican a lado y lado los votos de entrada y salida de cada nodo. Gracias a las propiedades homomórficas del algoritmo con el que se han cifrado los votos, el nodo puede hacer una prueba de conocimiento nulo que demuestre que el resultado de multiplicar los votos en su salida es el recifrado o descifrado del resultado de multiplicar los votos a su entrada. Esta propuesta contempla un mecanismo de cifrado en el que se puede detectar si la mixnet se ha aprovechado de las propiedades homomórficas, multiplicando un voto por un valor y dividiendo otro por ese mismo valor para así modificar el contenido de estos votos individuales sin que las pruebas dejen de funcionar.

- En [18] se seleccionan de forma aleatoria grupos de votos a la entrada de cada nodo, de manera que el nodo ha de revelar cuál es el grupo de votos correspondiente a la salida para poder hacer el producto a lado y lado y generar una prueba de conocimiento nulo de recifrado/descifrado para cada grupo. Así, existe una identificación de la correspondencia a nivel de grupos de votos, no individual.

- La propuesta [54] es muy semejante a este último esquema, pero el sistema de generación aleatoria de grupos que se propone mejora la eficiencia.

Se habla de propuestas de verificación heurística porque, dependiendo de cómo se generen las pruebas que relacionan votos de entrada y de salida para mantener la privacidad, la mixnet tiene una cierta probabilidad de hacer trampas y modificar votos sin ser detectada, aunque esa probabilidad es muy reducida.

Algunas de estas propuestas se habían roto hace un tiempo (como [35, 67]), es decir, se han encontrado ataques para los que la probabilidad de la mixnet de engañar no era tan pequeña como se creía. Pero recientemente Douglas Wikström ha publicado dos artículos en que propone ataques para otros esquemas de este tipo, uno para el Random Partial Checking [41] y otro para la propuesta [54], en [40]. Aunque no es fácil llevar a la práctica los ataques propuestos, (se tendrían que poner de acuerdo un volumen considerable de votantes y, al menos, el primer nodo de la mixnet), estas publicaciones han afectado de forma considerable a la credibilidad de los sistemas de verificación heurística.

Las propuestas basadas en sistemas de verificación heurística surgieron como contrapartida a las **propuestas de verificación exhaustiva**, porque éstas, para dar un cierto nivel de certeza más alto, de que la mixnet no puede mentir, necesitan de la generación de pruebas criptográficas muy costosas e incluso inasumibles en el caso de gran volumen de votos. Hasta hace unos años, la propuesta de verificación exhaustiva más eficiente era la de Andy Neff [48]. Pero recientemente, el trabajo de Jens Groth [14] y de Douglas Wikström [65, 68] en este tema, ha proporcionado muchas mejoras en la eficiencia de estos sistemas.

En este apartado sólo hemos hablado de la verificación del proceso de mixing, pero hay que tener en cuenta que algunas de las herramientas que se han presentado para los esquemas de recuento homomórfico, como el bulletin board para publicar los votos recibidos en el servidor de voto, y la generación y publicación de pruebas de descifrado o descifrado parcial cuando los miembros de la mesa descifran los votos, también se pueden aplicar en escenarios de votación donde se usan mixnets para proteger la privacidad de los votantes, para permitir a votantes y observadores verificar que los votos han sido emitidos por votantes válidos y que el proceso de descifrado ha sido correcto.

4.5. Recibos de votación: verificación counted-as-cast o recorded-as-cast

Los recibos de votación son unos valores que se generan a partir de un voto, una vez recibido en el servidor de voto, se devuelven al votante, y

se utilizan para que el votante pueda buscar este valor dentro de una lista para comprobar que su voto está en el servidor de voto (ha llegado) hasta una determinada fase del proceso. Dependiendo del tipo de recibo, la verificación se aplica en diferentes fases:

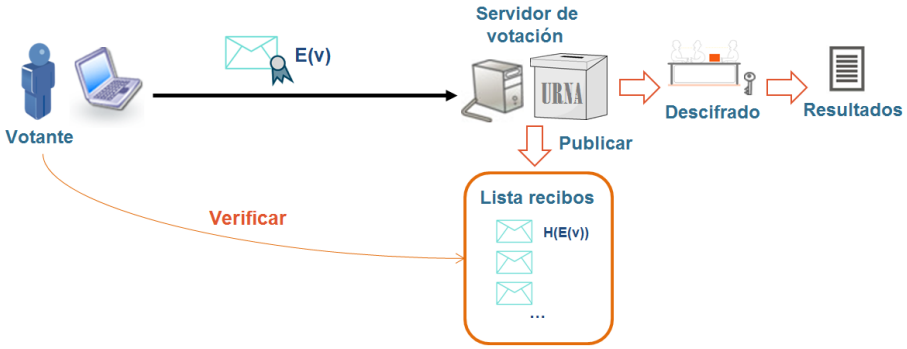

Figura 15: Generación y comprobación del recibo basado en el valor del voto cifrado.

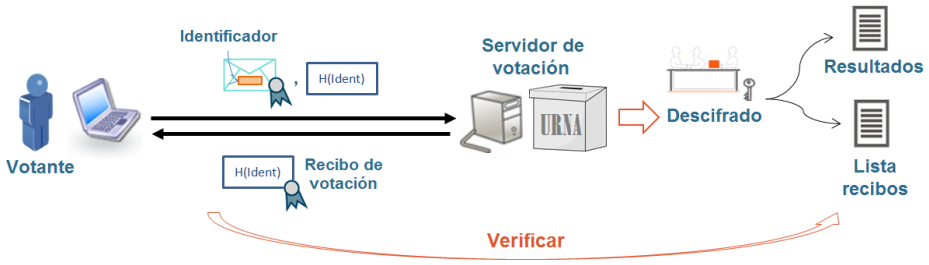

Figura 16: Generación y comprobación del recibo basado en un identificador del voto.

- Recibo basado en el valor del voto cifrado [34] (ver figura 15): este recibo suele generarse en el servidor del voto como el hash del voto cifrado que se recibe. Este valor se firma digitalmente con una clave del servidor de voto y se devuelve al votante. Una vez acabada la fase de voto, se publican en el bulletin board los valores de los recibos correspondientes a los votos almacenados en el servidor de voto que pasarán a la fase de recuento. Con este tipo de recibo, el votante puede hacer una verificación recorded-as-cast.

- Recibo basado en un identificador del voto [57] (ver figura 16): este recibo consiste en un número aleatorio (identificador) generado por el cliente de voto y cifrado conjuntamente con el voto. El voto cifrado y el valor de hash del identificador se envían al servidor de voto remoto, que firma digitalmente este segundo y lo envía de vuelta al votante. Una vez se descifran los votos, los valores de los identificadores que estaban cifrados conjuntamente con los votos, se publican en el bulletin board. Con este tipo de recibo, el votante puede hacer una verificación del tipo counted-as-cast, ya que puede saber que su voto ha llegado hasta la fase de descifrado.

5. Referencias básicas sobre criptografía

- Criptografía Simétrica,
  https://es.wikipedia.org/wiki/Criptografia_simetrica
- Criptografía Asimétrica,
  https://es.wikipedia.org/wiki/Criptografia_asimetrica
- Firma Digital,
  https://es.wikipedia.org/wiki/Firma_digital
- Criptosistema RSA,
  https://es.wikipedia.org/wiki/RSA
- Criptosistema ElGamal,
  https://es.wikipedia.org/wiki/Cifrado_ElGamal
- Compartición de secretos,
  https://es.wikipedia.org/wiki/Esquema_de_Shamir

Referencias

[1] —. “Dificultades para votar por correo debido al retraso en el envío de las papeletas”. Online: http://www.diariodenavarra.es/noticias/mas_actualidad/nacional/dificultades_para_votar_por_correo_debido_retraso_envio_las_papeletas_52492_1031.html 2

[2] —. “PKCS #1: RSA Cryptography Standard”. Online: http://www.rsa.com/rsalabs/node.asp?id=2125 8

[3] —. “PKCS #5: Password-Based Cryptography Standard”. Online: http://www.rsa.com/rsalabs/node.asp?id=2127 9

[4] —. “PKCS #12: Personal Information Exchange Syntax Standard”. Online: http://www.rsa.com/rsalabs/node.asp?id=2138 9

[5] —. “TPM”. Online: http://www.trustedcomputinggroup.org/developers/trusted_platform_module 27

[6] —. “Trusted Computing Group”. Online: http://www.trustedcomputinggroup.org/ 27

[7] —. “Uniformed and Overseas Citizens Absentee Voting Act”. Online: http://www.justice.gov/crt/spec_topics/military/uocava.php 2

[8] —. “What the Move Act Means For You”. Online: https://www.overseasvotefoundation.org/node/282 2

[9] —. “Wombat Voting”. Online: http://www.wombat-voting.com 31

[10] B. Adida. “Advances in Cryptographic Voting Systems”. Tesis Doctoral. Massachusetts Institute of Technology Cambridge, MA, USA, 2006. 1, 5, 17, 37

[11] B. Adida. “Helios: web-based open-audit voting”. En Proc. 17th Conference on Security Symposium (SS ’08), 335–348, USENIX Association, Berkeley, CA, USA, 2008. 29

[12] B. Adida, O. De Marneffe, O. Pereira, J. J. Quisquater. “Electing a university president using open-audit voting: analysis of real-world use of Helios”. En Proc. 2009 conference on Electronic voting technology/workshop on trustworthy elections (EVT/WOTE ’09), 10–10, USENIX Association, Berkeley, CA, USA, 2009. 29

[13] J. Algesheimer, J. Camenisch, V. Shoup. “Efficient computation modulo a shared secret with application to the generation of shared safe-prime products”. En Proc. CRYPTO ’02, Lecture Notes in Computer Science **2442**, 417–432. Springer, 2002. 8

[14] S. Bayer, J. Groth. “Efficient zero-knowledge argument for correctness of a shuffle”. En Proc. EUROCRYPT ’12, Lecture Notes in Computer Science **7237**, 263–280. Springer, 2012. 39

[15] J. Benaloh. “Simple verifiable elections”. En Proc. USENIX/Accurate Electronic Voting Technology Workshop 2006 (EVT ’06), 5–5, USENIX Association, Berkeley, CA, USA, 2006. Online: `http://www.usenix.org/event/evt06/tech/full_papers/benaloh/benaloh.pdf`. 28, 29

[16] J. C. Benaloh, M. Yung. “Distributing the power of a government to enhance the privacy of voters”. En Proc. 5th Annual ACM Symposium on Principles of Distributed Computing (PODC ’86), 52–62, ACM, New York, NY, USA, 1986. 14

[17] D. Boneh, M. K. Franklin. “Efficient generation of shared rsa keys”. En Proc. CRYPTO ’97, Lecture Notes in Computer Science **1294**, 425–439. Springer, 1997. 8

[18] D. Boneh, P. Golle. “Almost entirely correct mixing with applications to voting”. En Proc. 9th ACM Conference on Computer and Communications Security (CCS ’02), 68–77, ACM, New York, NY, USA, 2002. 38

[19] D. Chaum. “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms”. Communications of the ACM **24(2)**, 84–90, 1981. 12, 18

[20] D. Chaum. “Blind Signatures for Untraceable Payments”. En Proc. CRYPTO ’82, 199–203, ISBN 978-1-4757-0602-4. Springer, 1983. 13

[21] D. Chaum. “Secret-Ballot Receipts and Transparent Integrity. Better and Less-costly Electronic Voting at Polling Places”. Online: `http://www.vreceipt.com/article.pdf`, 2003. 38

[22] M. R. Clarkson. “Civitas: Toward a Secure Voting System”. En Proc. 2008 IEEE Symposium on Security and Privacy (SP ’08), 354–368, IEEE Computer Society, Washington, DC, USA, 2008. 25

[23] R. Cramer, R. Gennaro, B. Schoenmakers. "A secure and optimally efficient multi-authority election scheme". En Proc. EUROCRYPT '97, Lecture Notes in Computer Science **1233**, 103–118. Springer, 1997. 14, 36

[24] I. Damgård, K. Dupont. "Efficient threshold rsa signatures with general moduli and no extra assumptions". En Proc. PKC '05, Lecture Notes in Computer Science **3386**, 346–361. Springer, 2005. 8

[25] I. Damgård, M. Koprowski. "Practical threshold rsa signatures without a trusted dealer". En Proc. EUROCRYPT '01, Lecture Notes in Computer Science **2045**, 152–165. Springer, 2001. 8

[26] I. Damgård and G. L. Mikkelsen. "Efficient, robust and constant-round distributed rsa key generation". En Proc. TCC '10, Lecture Notes in Computer Science **5978**, 183–200. Springer, 2010. 8

[27] R. Dingledine, N. Mathewson, P. Syverson. "Tor: The Second-Generation Onion Router". En Proc. 13th USENIX Security Symposium, (SSYM '04), 21–21, USENIX Association, Berkeley, CA, USA, 2004. 10

[28] T. ElGamal. "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". IEEE Transactions on Information Theory **31(4)**, 469–472, 1985. 15

[29] A. Escala, C. Luna, S. Guasch. "Implementación de la generación y firma RSA distribuida en procesos de voto electrónico". En Proc. XI Reunión española sobre criptología y seguridad de la información (RECSI '10), 117–122, ISBN: 978-84-693-3304-4, 2010. 8

[30] Y. Frankel, P. D. MacKenzie, M. Yung. "Robust efficient distributed RSA-key generation". En Proc. 30th annual ACM symposium on Theory of Computing (STOC '98), 663–672, ACM, New York, NY, USA, 1998. 8

[31] A. Fujioka, T. Okamoto, K. Ohta. "A practical secret voting scheme for large scale elections". En Proc. AUSCRYPT '92, Lecture Notes in Computer Science **718**, 244–251. Springer, 1993. 12

[32] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". Journal of Cryptology **20(1)**, 51–83, 2007. 8

[33] R. Gharadaghy, M. Volkamer. “Verifiability in electronic voting - explanations for non security experts”. En Proc. EVOTE ’10, Lecture Notes in Informatics **167**, 151–162. Springer, 2010. 27

[34] K. Gjøsteen. “Analysis of an internet voting protocol”. Cryptology ePrint Archive: Report 2010/380. 2010. Online: `http://eprint.iacr.org/2010/380`. 33, 40

[35] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, A. Juels. “Optimistic Mixing for Exit-Polls”. En Proc. ASIACRYPT ’02, Lecture Notes in Computer Science **2501**, 451–465. Springer, 2002. 38, 39

[36] M. Hirt. “Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting”. Tesis Doctoral. ETH Zurich. Reprint as ETH Series in Information Security and Cryptography **3**. ISBN 3-89649-747-2. Hartung-Gorre Verlag, Konstanz, 2001. 28

[37] M. Jakobsson, A. Juels. “Millimix: Mixing in Small Batches. Technical Report”. Center for Discrete Mathematics & Theoretical Computer Science, 1999. 24

[38] M. Jakobsson, A. Juels, R. L. Rivest. “Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking”. En Proc. 11th USENIX Security Symposium, 339–353, USENIX Association, Berkeley, CA, USA, 2002. 38

[39] A. Juels, D. Catalano, M. Jakobsson. “Coercion-resistant electronic elections”. En Proc. 2005 ACM workshop on Privacy in the electronic society (WPES ’05), 61–70, ACM, New York, NY, USA, 2005. 23, 25

[40] S. Khazaei, B. Terelius, D. Wikström. “Cryptanalysis of a Universally Verifiable Efficient Re-encryption Mixnet”. Cryptology ePrint Archive: Report 2012/10. 2012. Online: `http://eprint.iacr.org/2012/10`. 39

[41] S. Khazaei, D. Wikström. “Randomized Partial Checking Revisited”. Cryptology ePrint Archive: Report 2012/63. 2012. Online: `http://eprint.iacr.org/2012/63`. 39

[42] A. Kiayias, M. Yung. “The Vector-Ballot E-Voting Approach”. En Proc. FC ’04, Lecture Notes in Computer Science **3110**, 72–89. Springer, 2004. 21

[43] R. Koenig, R. Haenni, S. Fischli. “Preventing Board Flooding Attacks in Coercion-Resistant Electronic Voting Schemes”. En Proc. Future Challenges in Security and Privacy for Academia and Industry (SEC ’11). IFIP Advances in Information and Communication Technology **354**, 116–127, Springer, 2011. 25

[44] L. Langer, A. Schmidt, M. Volkamer, J. Buchmann. “Classifying Privacy and Verifiability Requirements for Electronic Voting”. En Proc. INFORMATIK 2009, Im Focus das Leben, Lecture Notes in Informatics, GI Editions, **P-154**, 1837–1846, 2009. 30

[45] D. Malkhi, O. Margo, E. Pavlov. “E-Voting Without ‘Cryptography’”. En Proc. FC ’02, Lecture Notes in Computer Science **2357**, 1–15. Springer, 2002. 10

[46] M. Marlinspike. “New Tricks For Defeating SSL In Practice”. Blackhat conference, Washington, DC, USA, 2009. 4

[47] A. J. Menezes, S. A. Vanstone, P. C. Van Oorschot. “Handbook of Applied Cryptography”. (1st ed.), ISBN: 08-493-8523-7. CRC Press, Inc., Boca Raton, FL, USA. 1996. 8

[48] C. A. Neff. “A verifiable secret shuffle and its application to e-voting”. En Proc. 8th ACM conf. on Computer and Communications Security (CCS ’01), 116–125, ACM, New York, NY, USA, 2001. 39

[49] M. Olembo, P. Schmidt, M. Volkamer. “Introducing Verifiability in the POLYAS Remote Electronic Voting System”. En Proc. Sixth International Conference on Availability, Reliability and Security (ARES ’11), 127–134, IEEE Computer Society, Washington, DC, USA, 2011. 12

[50] P. Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. En Proc. EUROCRYPT ’99, Lecture Notes in Computer Science **1592**, 223–238. Springer, 1999. 15

[51] K. Peng. “A Hybrid E-Voting Scheme”. En Proc. ISPEC ’09, Lecture Notes in Computer Science **5451**, 195–206. Springer, 2009. 20

[52] K. Peng, F. Bao. “Batch ZK Proof and Verification of OR Logic”. En Proc. ISC ’09, Lecture Notes in Computer Science **5487**, 141–156. Springer, 2009. 21

[53] K. Peng, F. Bao. “Efficient multiplicative homomorphic e-voting”. En Proc. ISC ’10, Lecture Notes in Computer Science **6531**, 381–393. Springer, 2011. 20

[54] J. Puiggalí, S. Guasch. "Universally Verifiable Efficient Re-encryption Mixnet". En Proc. EVOTE '10, Lecture Notes in Informatics **167**, 241–254. Springer, 2010. 38, 39

[55] J. Puiggalí, S. Guasch. "Internet Voting System with Cast as Intended Verification". En Proc. VoteID '11, Lecture Notes in Computer Science **7187**, 36–52. Springer, 2012. 33

[56] J. Quisquater, L. C. Guillou, T. A. Berson. "How to Explain Zero-Knowledge Protocols to Your Children". En Proc. CRYPTO '89, Lecture Notes in Computer Science **435**, 628–631. Springer, 1989. 16

[57] A. Riera. "Design of Implementable Solutions for Large Scale Electronic Voting Schemes". Tesis Doctoral. ISBN 84-490-1810-2, Publication Services of the UAB, Bellaterra, 1999. 37, 41

[58] R. Rivest, A. Shamir, L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM **21(2)**, 120–126, 1978. 8

[59] K. Sako, J. Kilian. "Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth". En Proc. EUROCRYPT '95, Lecture Notes in Computer Science **921**, 393–403. Springer, 1995. 37

[60] D. Sandler, A. Derr, D. S. Wallach. "VoteBox: a tamper-evident, verifiable electronic voting system". En Proc. 17th Conference on Security Symposium (SS '08), 349–364, USENIX Association, Berkeley, CA, USA, 2008. 31

[61] B. Schneier. "Applied Cryptography: Protocols, Algorithms, and Source Code in C". (2nd ed.), ISBN: 978-0-471-11709-4. 1995. 11, 13

[62] A. Shamir. "How to share a secret". Communications of the ACM **22(11)**, 612–613, 1979. 7

[63] O. Spycher, R. Koenig, R. Haenni, M. Schläpfer. "A new approach towards coercion-resistant remote e-voting in linear time". En Proc. FC '11, Lecture Notes in Computer Science **7035**, 182–189. Springer, 2012. 25

[64] T. W. Storer. "Practical Pollsterless Remote Electronic Voting". Tesis Doctoral. The University of St Andrews, UK, 2007. 10

[65] B. Terelius, D. Wikström. “Proofs of Restricted Shuffles”. En Proc. AFRICACRYPT ’10, Lecture Notes in Computer Science **6055**, 100–113. Springer, 2010. 39

[66] S. Vadhan, A. Rosen. “Public-Key Encryption in Practice”. Harvard University. Introduction to Cryptography, Lecture Notes 15, 2006. 13

[67] D. Wikström. “Five practical attacks for optimistic mixing for exit-polls”. En Proc. SAC ’03, Lecture Notes in Computer Science **3006**, 160–175. Springer, 2004. 39

[68] D. Wikström. “A Commitment-Consistent Proof of a Shuffle”. Cryptology ePrint Archive: Report 2011/168. 2011. Online: `http://eprint.iacr.org/2011/168`. 39