Implementación de un Protocolo Confidencial y Multiparte para Entregas Certificadas sobre la Blockchain Ethereum

Macià Mut-Puigserver
Dpt. de Ciències Matemàtiques i Informàtica
Universitat de les Illes Balears
Ctra. de Valldemossa, km 7,5. E-07122 Palma, Spain
macia.mut@uib.cat

Miquel A. Cabot-Nadal
Dpt. de Ciències Matemàtiques i Informàtica
Universitat de les Illes Balears
Ctra. de Valldemossa, km 7,5. E-07122 Palma, Spain
miquel.cabot@uib.cat

M. Magdalena Payeras-Capellà
Dpt. de Ciències Matemàtiques i Informàtica
Universitat de les Illes Balears
Ctra. de Valldemossa, km 7,5. E-07122 Palma, Spain
mpayeras@uib.cat

Llorenç Huguet Rotger
Dpt. de Ciències Matemàtiques i Informàtica
Universitat de les Illes Balears
Ctra. de Valldemossa, km 7,5. E-07122 Palma, Spain
l.huguet@uib.cat

***Resumen*—Recientemente se han presentado varias propuestas de soluciones basadas en blockchain para aplicaciones tradicionales de comercio electrónico, aprovechando el hecho de que blockchain es una tecnología que ofrece un registro inmutable de datos. Entre estas propuestas podemos encontrar soluciones para notificaciones certificadas, firma digital de contratos, protocolos de custodia, pagos justos y entregas registradas. Para ejecutar intercambios justos, la mayoría de las soluciones involucran a terceros confiables, conocidos como TTP, que supervisan los intercambios para garantizar algunas propiedades de seguridad. Los servicios de entrega electrónica registrada (eDelivery) permiten al usuario probar que ha enviado algunos datos a un conjunto de receptores. En este artículo presentamos una implementación de un protocolo que logra las mejores propiedades de soluciones anteriores al mismo tiempo. El nuevo protocolo no requiere la participación de una TTP en ningún momento, mientras permite la entrega electrónica de datos confidenciales, satisfaciendo los requisitos de seguridad para este servicio.**

***Index Terms*—Blockchain, Notification, Smart Contract, Confidentiality, Fairness, Cryptocurrencies.**

I. Introducción

Existen diferentes soluciones basadas en blockchain que, aprovechando el hecho de que es una tecnología que ofrece un registro de datos inmutable, solucionan aplicacciones tradicionales de comercio electrónico: notificaciones certificadas, firma digital de contratos, compras atómicas, entregas certificadas, etc. En [6] se presentaron dos propuestas basadas en blockchain para los servicios de entrega electrónica registrada o certificada (eDelivery). Este servicio proporciona a los usuarios la prueba de que el remitente ha enviado unos datos y una verificación electrónica de que los datos fueron entregados o que se hizo un intento de entrega (está disponible para el receptor). Los servicios de entrega registrados son ofrecidos principalmente por servicios postales en muchos países y tienen diferentes denominaciones dependiendo de cada proveedor de servicios. Por ejemplo, la mayoría de los servicios postales ofrecen un servicio de correo registrado que incluye el envío de correspondencia, documentos y objetos de valor. En el caso del Servicio Postal de los Estados Unidos (USPS), el servicio se llama Correo certificado, pero también ofrece un servicio de Correo registrado que además proporciona las propiedades de la cadena de custodia. Es decir, la recopilación de información que proporciona evidencia sobre las acciones cronológicas en el servicio de entrega o secuencia de custodia, control, transferencia, análisis y disposición de la entrega. Además, el servicio de correo registrado de USPS puede especificar el estado de entrega o el estado de intento de entrega cuando el artículo llega a su destino [1]. De esta manera, este tipo de servicios proporciona evidencia de que un usuario que actúa como receptor (o conjunto de receptores) tiene acceso a los datos desde un instante específico. Como servicios de confianza, estas propuestas deben ofrecer un alto nivel de seguridad y protección de la privacidad de los usuarios, pero también deben tener en cuenta las regulaciones sobre el tema. Por lo tanto, las características de las tecnologías blockchain hacen de ella una herramienta ideal para ofrecer confianza y seguimiento de datos para nuevas soluciones de entrega electrónica.

Es habitual que los usuarios de comercio electrónico intercambien datos o elementos entre ellos. Un intercambio justo tiene como objetivo proporcionar un trato igualitario a todas las partes involucradas. Al final de la ejecución de un protocolo, cada parte tiene el elemento que desea obtener de la otra parte involucrada o, si no es el caso, el intercambio no se ha realizado con éxito para ningún usuario, es decir, ningún usuario recibió el elemento deseado. En el diseño de estos protocolos se requiere un método que permita realizar los intercambios y garantizar la seguridad del intercambio.

Para ejecutar intercambios equitativos, la mayoría de las soluciones incluyen terceras partes de confianza que administran los intercambios con más o menos participación. Las TTP son responsables de la resolución de todos los conflictos que puedan surgir entre las partes como resultado de un intercambio no concluido o un intento de fraude. Los

[1](https://faq.usps.com/s/article/What-is-Registered-Mail#internacional)

protocolos existentes de intercambio equitativo, como [4], [5], [16] involucran a una TTP en varios grados, con funciones similares a las de un juez o notario. Sin embargo, la aceptación de las TTPs puede ser un obstáculo para generalizar el uso de este tipo de protocolos. Primero, es difícil tener TTPs realmente confiables para cualquier usuario y tenemos que tener en cuenta que deben ser útiles en diferentes escenarios (por ejemplo, los documentos electrónicos generados por una TTP deben ser aceptados para resolver disputas en tribunales de diferentes países). Por lo tanto, las TTP podrían causar problemas técnicos (por ejemplo, cuellos de botella), reducir la eficiencia de los protocolos (por ejemplo, retrasos en la resolución de conflictos) y también aumentar los costes de ejecución (por ejemplo, altas tasas de servicio). Además, son un punto muy sensible porque la seguridad del intercambio podría verse comprometida si la TTP tiene alguna vulnerabilidad. En los protocolos presentados en [6] para eDelivery, los elementos a intercambiar son los datos que se entregarán junto con las pruebas de no repudio de origen y recepción. Los dos protocolos presentados en [6] difieren en las propiedades logradas y también en el uso de TTPs. El primer protocolo es una solución basada en blockchain sin TTP para la entrega electrónica de datos no confidenciales. El segundo protocolo permite también la entrega electrónica de datos confidenciales. Sin embargo, esta segunda propuesta requiere la participación de una TTP en una fase de resolución no obligatoria. Desde [6] no se han propuesto nuevos protocolos para eDelivery confidencial sin TTP.

En este artículo presentamos un nuevo protocolo que logra las mejores propiedades de las soluciones anteriores al mismo tiempo, evitando la necesidad de elegir una propiedad y renunciar a la otra. No requiere la participación de una TTP en ningún momento mientras permite la entrega electrónica de datos confidenciales.

Se han considerado diversas herramientas criptograficas para ofrecer anonimato. En primer lugar se ha implementado la aplicación utilizando pruebas de conocimiento nulo basadas en Schnorr. Posteriormente, y por razones de eficiencia y coste se ha rediseñado el protocolo para el uso de criptografia de curva elíptica.

II. Visión general del sistema

El sistema propuesto aquí ofrece confidencialidad y un intercambio equitativo para ambas partes. Es decir, el contenido de la notificación se mantiene confidencial y se garantiza el intercambio del mensaje a cambio de las correspondientes pruebas de no repudio de recepción. El protocolo consigue mantener esta equidad sin que el contrato inteligente tenga acceso al contenido de los datos entregados y no se tenga que registrar el mensaje en claro en la cadena de bloques. El remitente, $Alice$, ejecuta el primer paso del protocolo mediante la DApp para registrar los datos cifrados de la notificación en la cadena de bloques. El cifrado debe garantizar que solo los receptores puedan acceder al contenido de la notificación. Además, debido a la naturaleza del servicio de notificación multiparte, este paso debe diseñarse de manera que el contrato inteligente ejecute una verificación para garantizar que el mensaje que cada receptor puede descifrar sea el mismo. Para ejecutar este paso, todos los usuarios deben generar un par de claves, que se denominarán claves de notificación. Los receptores, miembros del conjunto R, deben aceptar la notificación por medio de una transacción. La transacción se almacena en la cadena de bloques. Finalmente, $Alice$ ejecutará una nueva transacción finalizando el intercambio en el tercer paso del protocolo y dando acceso al contenido del mensaje a los receptores que han aceptado el intercambio.

III. Protocolo Confidencial y Multiparte para Entregas Certificadas sobre Blockchain sin TTP

III-A. Antecedentes criptográficos y notación

Antes de iniciar el uso del servicio de notificaciones que proponemos, se tienen que publicar los correspondientes parámetros criptográficos para asegurar la seguridad del sistema. De la misma forma, los usuarios tienen que poder comprobar la corrección de estos. Para una configuración apropiada de las condiciones operacionales del protocolo, antes del inicio de la fase de *Creation* de la notificación por parte de $Alice$, tienen que establecerse de forma apropiada los siguientes parámetros:

- F_p: cuerpo finito de p elementos, donde p es un número primo
- $E(F_p)$: curva elíptica sobre F_p
- Sea G un generador de un subgrupo cíclico de puntos sobre $E(F_p)$ de orden n
- n: orden de G
- h: cofactor del subgrupo generado por G, que es igual al orden de la curva elíptica dividido por el orden del subgrupo cíclico n (normalmente $\leqslant 4$)
- Denotamos por $P\mathrm{x}[b]$ la multiplicación de un punto P por un escalar b sobre $E(F_p)$
- Sea $a \leftarrow [1, n-1]$ la clave privada de $Alice$ que ha sido generada de forma aleatoria entre $[1, n-1]$
- Clave pública de $Alice$: $A = G\mathrm{x}[a]$
- Los usuarios que interactúan con $Alice$ tienen que verificar que A es un punto válido de la curva y que $A\mathrm{x}[h]$ no es un punto en el infinito

III-B. Fases del Protocolo

El protocolo de intercambio equitativo entre la emisión de una notificación y el intercambio de evidencias de origen por evidencias de recepción del mensaje consta de tres fases. Hemos etiquetado estas etapas con los nombres de *Creation*, *Accept* y *Finish*. Además, incluimos una cuarta fase, *Cancellation*, que opcionalmente pueden ejecutar los receptores de las notificaciones. Describimos estas fases a continuación (en la Tabla I se encuentra la notación utilizada en la especificación de los protocolos):

1. **Creation** Subprotocolo 1. $Alice$ genera su par de claves (a, A) y genera la semilla aleatoria v que utiliza para cifrar el mensaje M. Si es necesario, para esta operación de cifrado puede fragmentarse el contenido de la notificación ($M[j]$). Para iniciar la emisión de la notificación, $Alice$ crea una nueva instancia del smart contract invocando a la función $creation()$ del constructor de la factoría del smart contract desplegado por el proveedor del servicio. Esta nueva instancia servirá para administrar la nueva entrega. La llamada a la función $creation()$ incluye los siguientes parámetros: el conjunto de receptores de

Tabla I: Notación para el protocolo de entrega certificada basado en blockchain.

Notación	
$Alice$	Emisor.
R	Conjunto de receptores. Bob_i denota un receptor individual.
R_a	Conjunto de receptores que ha aceptado la entrega.
M	Mensaje, contenido de la entrega.
X, Y	Concatenación de messages X y Y.
$U \blacktriangleright e.f$	Ejecución de la función f de e por parte de U.
t_1	Timeout para que Bob_i acepte la entrega.
t_2	Timeout para que $Alice$ finalice el intercambio.
D	Deposito enviado al Smart Contract (ethers).
a	Clave privada de notificación de Alice.
A	Clave pública de notificación de Alice.
b_i	Clave privada de notificación de Bob_i.
B_i	Clave pública de notificación de Bob_i.
G, p, n	Parámetros Criptográficos del sistema.
v	Nonce secreto de cifrado utilitzado por $Alice$.
s_i	Nonce secreto de cifrado utilitzado por Bob_i.
$V = G\text{x}[r]$	Primer elemento computado en la prueba ZKP.
$M[i]$	Fragmentos del mensaje M que se va a cifrar, en el rango $0 < M_i < p$
$hash()$	Función de Hash.
$key = random.seed(hash(v))$	Clave de cifrado.
$C[i] = M[i]$ XOR key	Fragmento i del cifrado del mensaje.
c_i	Reto enviado por Bob_i a $Alice$.
Z_{i1}=G$\text{x}s_i$	Primera parte del cifrado de la clave secreta de notificación del receptor.
$Z_{i_2} = (A\text{x}[s_i]) \oplus (b_i)$	Segunda parte del cifrado de la clave secreta de notificación del receptor.
r_i	Respuesta al reto c_i.

la notificación R, el compromiso de $Alice$ de enviar la clave correcta V, el criptograma C, los plazos del envío $\{t_1, t_2\}$ y el resto de parámetros criptográficos. El primer plazo t_1 sirve para determinar el periodo de aceptación del envío que tienen los receptores, mientras que el segundo plazo especifica el tiempo que tiene el emisor para finalizar el envío de la notificación y, por lo tanto, determina el tiempo a partir del cual los receptores ya tendrán a su disposición el contenido del mensaje junto con la prueba de no repudio de origen. Si en t_2 el emisor no ha completado el envío, los receptores que lo deseen pueden obtener una prueba de cancelación invocando a la función $cancel()$ del smart contract. Opcionalmente, en esta etapa se puede incluir un deposito D o pago por el servicio. Notar que se ha hecho una adaptación del método de cifrado Elliptic Curve Integrated Encryption Scheme (ECIES) [2], [3], donde el elemento V no es utilizado por el receptor para descifrar el mensaje si no que V es utilizado como compromiso de $Alice$ de emisión de la clave correcta que será comprobada por el smart contract a través de una prueba de conocimiento nulo no-interactiva de Schnorr [1] en la tercera fase de este protocolo ($Finish$).

Subprotocolo 1: Paso 1. Creation

1. $Alice$: generates:
   $M, a \leftarrow [1, n-1], A = G\text{x}[a], v \leftarrow [1, n-1]$
2. $Alice$: $key = random.seed(hash(v))$
3. $Alice$: encryption of C.
   If required, fragmentation of M in blocks.
   $V = G\text{x}[v]$
   FOR $j = 1$ **TO** $M.length$
   $C[j] = M[j]$ XOR key
   $key = random.seed(hash(key))$
4. $Alice \blacktriangleright$
   $SM.creation(Alice, R, V, C, t_1, t_2, A, G, p, n, D)$
5. SM : $State_i = Created, \forall i$

2. **Accept** Subprotocolo 2. En un escenario de envío de notificación con múltiples destinatarios, cada receptor decide individualmente si acepta o no la recepción de esta. Para aceptar la entrada de la nueva notificación,

Bob_i tiene que ejecutar la correspondiente función del smart contract que se ha desplegado a tal efecto antes de la deadline t_1. De esta forma, si un determinado receptor Bob_i no la acepta antes de t_1, se asume que no quiere recibirla y el estado de la notificación para este receptor pasará a ser $State_i = Rejected$. En caso contrario, la recepción del mensaje por parte de Bob_i ha sido aceptado y el estado de esta notificación para este receptor será $State_i = Accepted$.
La emisora $Alice$ tiene que permitir el acceso al contenido del mensaje a todos aquellos miembros de R que han aceptado la entrega, manteniendo oculto el contenido para el resto. Por lo tanto, el mensaje en claro no puede ser incluido en una transacción ni tampoco puede ser guardado en la blockchain. Esto significa que $Alice$ tiene que generar los elementos necesarios para garantizar la entrega confidencial de la notificación y el protocolo debe asegurarnos que la clave de descifrado para que todos los usuarios de R_a sea la misma (para que todos puedan leer el mismo contenido). Es decir, $Alice$ tiene que enviar la clave a la que se ha comprometido durante la fase de $Creation$ con el parámetro V. El smart contract tiene que asegurar que todos los receptores en R_a tendrán acceso al mismo contenido descifrado de la notificación. Con este objetivo, en esta fase, cada receptor Bob_i genera su par de claves de notificación (b_i, B_i) y envía su clave pública de notificación B_i, junto con un nonce c_i que funciona como reto para la ZKP a través de la cual el smart contract podrá comprobar que todos los receptores reciben la clave que $Alice$ se había comprometido con V. Además, el receptor Bob_i cifra su clave privada $\{Z_{i_1}, Z_{i_2}\}$ con la clave pública de $Alice$ para que ella pueda recuperarla posteriormente y, durante la fase de $Finish$, pueda enviarle la clave a Bob_i para abrir el contenido de la notificación.

Subprotocolo 2: Paso 2. Accept

1. Bob_i : generation: $b_i \leftarrow [1, n-1], B_i = G\text{x}[b_i], s_i$
2. Bob_i :
   $Z_{i_1} = G\text{x}[s_i], Z_{i_2} = (A\text{x}[s_i]) \oplus (b_i), c_i \leftarrow [1, n-1]$
3. Bob_i ► $SM.accept(Z_{i_1}, Z_{i_2}, B_i, c_i)$
2. SM:
   IF$(now < t_1)$ AND $(Id == Bob_i)$ AND
   $(State_i == Created)$
   $State_i = Accepted$
   Add Bob_i to R_a

3. **Finish** Subprotocolo 3. Finalmente, antes del plazo definido por t_2, $Alice$ puede finalizar el proceso de entrega para aquellos B_i que hayan aceptado la notificación, ejecutando la función $finish()$ del smart contract.
   En esta fase, el emisor $Alice$ genera para cada receptor en R_a, es decir, los receptores que han aceptado la entrega, una respuesta al desafío en forma de ZKP utilizando el elemento secreto v empleado para cifrar el mensaje en el primer paso del protocolo, el desafío proporcionado en la fase 2 (c_i) y la clave de notificación compartida secreta de B_i: b_i. Se ha de tener en cuenta que, aunque la clave secreta de notificación compartida, b_i, fue creada por B_i, B_i ha enviado la clave compartida secreta a $Alice$ encriptada con la clave pública de $Alice$, lo que resulta en $\{Z_{i1}, Z_{i2}\}$. De esta forma, $Alice$ puede obtener la clave secreta de notificación compartida para cada B_i (ver paso 1 del Subprotocolo 3).
   El Smart Contract almacenará el parámetro recibido (r_i) y verificará que cada receptor B_i en R_a tendrá acceso al mismo elemento secreto v para poder descifrar el mensaje. Sin embargo, el contrato inteligente no tiene conocimiento de v y, por lo tanto, el mensaje se mantiene confidencial. Es decir, la ZKP permite al smart contract verificar el compromiso de bits con la clave secreta v que hizo $Alice$ en la fase de creación de la eDelivery, de tal manera que $Alice$ puede probar al smart contract que está enviando el elemento secreto apropiado v a cada receptor sin revelar su valor, ya que su respuesta al desafío enviado por cualquier receptor c_i es coherente con el valor comprometido de la clave secreta expresado públicamente en V. Si la verificación es correcta, el smart contract da por finalizada de forma correcta la notificación.
   En el último paso de esta fase, los receptores pueden aislar v de r_i debido al conocimiento de b_i y, por lo tanto, podrán leer el contenido del mensaje. Finalmente, después de t_2, cada receptor en R_a puede acceder al mensaje a través de WEB3 o una interfaz similar o, en el caso de que el emisor no haya completado con éxito el protocolo, los receptores tendrán acceso al mensaje evidencia de cancelación correspondiente.

Subprotocolo 3: Paso 3. Finish

1. $Alice$: decrypts: $b_i = Z_{i_2} \oplus (Z_{i_1}\text{x}[a])$
2. $Alice$: computes: r_i=$v - b_i * c_{i\ mod n}$
3. $Alice$ ► SM.finish(r_i)
4. SM:
   IF $(Id == Alice)$ AND $((t_1 < now < t_2)$ OR
   $(R_a == R))$
   FOR $(\forall Bob_i \in R_a)$
   IF $V == G\text{x}[r_i] + A\text{x}[c_i]$
   $State_i = Finished$
   FOR $(\forall Bob_i \notin R_a)$
   $State_i = Rejected$
   Deposit D is refunded to $Alice$
5. Bob_i :
   $v = r_i + b_i$*$c_{i\ mod n}$
   $key = random.seed(hash(v))$
   FOR $j = 1$ **TO** n
   $M[j] = C[j]$ XOR key
   $key = random.seed(hash(key))$

4. **Cancellation** Subprotocolo 4. Cancelación de aceptación. Este paso es opcional y será ejecutado por cualquier receptor B_i, si el remitente $Alice$ no finaliza

el intercambio proporcionando la clave de descifrado cuando el receptor había aceptado el intercambio.

Subprotocolo 4: Cancellation of Acceptance

1. Bob_i ► SM.cancel()
2. SM:
   IF($now >= t2$, $Id == Bob_i$ AND
   $State_i == Accepted$)
   $State_i = Cancelled$

IV. Análisis de propiedades y Coste

En esta sección se presenta un análisis de seguridad sobre las propiedades deseadas para los sistemas de entrega electrónica certificada [6]: efectividad, equidad, transferencia de evidencias, temporalidad, no repudio y confidencialidad.

1. ***Efectividad.*** El sistema de entregas certificadas presentado en este documento es efectivo. Por lo tanto, todas las partes recibirán los elementos esperados en caso de comportarse de acuerdo con el protocolo. Para crear una nueva entrega, el remitente genera una nueva instancia del contrato inteligente para ejecutar las funciones siguiendo las especificaciones del protocolo. Si todas las partes ejecutan todas las funciones correctamente (funciones $accept()$ y $finish()$), tendrán los elementos que esperaban recibir. Esto puede deducirse fácilmente del protocolo presentado en III-B. Al final del intercambio, los receptores que han seguido el protocolo tendrán la clave para descifrar los datos entregados y la prueba de no repudio de origen, mientras que el remitente tendrá la prueba de repudio de los receptores que hayan finalizado el intercambio.

2. ***Equidad y Transferencia de evidencias***
   El protocolo propuesto es equitativo, al final de la ejecución de un protocolo, cada parte ha recibido el elemento adecuado o ninguna de las partes ha recibido datos útiles sobre el elemento de la otra parte, lo que proporciona una gran imparcialidad [7]. Además, la evidencia generada por el protocolo puede transferirse a una parte externa para probar el resultado y los efectos del intercambio, sin más verificaciones. Por un lado, de acuerdo con el protocolo, el remitente no recibirá la evidencia de no repudio de recepción generada por el contrato inteligente, excepto si ejecuta una transacción para permitir que los receptores descifren el mensaje y, al mismo tiempo permite que el contrato inteligente verifique la validez de la clave proporcionada (estado = $Finished$). Por otro lado, cualquier receptor B_i puede obtener acceso a los datos entregados, si ejecuta una transacción para aceptar recibir la entrega electrónica (estado = $Accepted$). Si las partes no siguen las especificaciones del protocolo, es decir, si no ejecutan las funciones $accept()$ y/o $finish()$, el contrato inteligente garantiza un resultado equitativo para todos los usuarios sin necesidad de intervención de una TTP:
   - $accept()$ no ejecutado. El contrato inteligente establecerá su estado en $Rejected$, después del tiempo de espera, para aquellos receptores que no hayan llamado a la función $accept()$. La entrega no se completará para estos receptores, por lo que no recibirán la clave de descifrado.
   - $finish()$ no ejecutado. Los receptores pueden llamar a la función $cancel()$ para concluir el intercambio con equidad, si el remitente no ha ejecutado la función $finish()$. De esta forma, el contrato inteligente establecerá el estado en $Cancelled$.

   Aunque el contrato inteligente puede crear pruebas de cancelación alternativas, el protocolo no permite ninguna circunstancia en la que un usuario pueda obtener evidencia contradictoria ya que el estado de cada receptor se actualiza en el contrato inteligente (por lo tanto, no es posible realizar una acción que pueda conducir a una situación de equidad débil [7]).

3. ***Temporalidad*** Los parámetros temporales a considerar son la asincronía y las marcas temporales. Para el protocolo presentado una entrega exitosa siempre se completará antes del plazo t_2. Si la entrega no es exitosa, tenemos diferentes plazos en función de cómo se realizó el intercambio. Si un receptor no acepta la entrega, la entrega se clasificará como *rejected* en el t_1. Si después de la aceptación de la entrega por parte del receptor, el remitente no finaliza el intercambio, entonces el intercambio puede cancelarse en el instante t_2. Además, el sistema solicita al remitente que finalice el intercambio para la entrega certificada antes del plazo t_2. Esta motivación se realiza mediante un depósito que está bloqueado en el contrato inteligente. El depósito se devolverá al remitente en caso de conclusión antes de t_2. Hay que tener en cuenta que el blockchain marca la hora de todas las transacciones realizadas en él, por lo el intercambio dispone de marcas temporales para todas las operaciones sobre el smart contract que generan transaccion, como las fases del protocolo propuesto.

4. ***No repudio*** El protocolo de entrega electrónica certificada debe proporcionar una evidencia de no repudio de recepción y también una evidencia de no repudio de origen. Con respecto al origen, el emisor de la entrega no puede negar haber ejecutado la función de creación ya que hay una transacción en la cadena de bloques desde su dirección que contiene las direcciones de los receptores y el mensaje cifrado (ver Subprotocolo 1). Además, la transacción relacionada con la ejecución de la función $finish()$ (ver Subprotocolo 3) prueba que el emisor ha proporcionado la clave para descifrar el mensaje a los receptores que aceptaron la entrega electrónica, y se considera el elemento de prueba de no repudio de origen ya que esta transacción lleva el intercambio al estado $Finished$.
   Con respecto a la recepción, cada destinatario B_i no puede rechazar haber aceptado la entrega porque existe una transacción de aceptación desde su dirección

almacenada en la cadena de bloques. Esta transacción acepta la recepción de la entrega y, de acuerdo con eso, el contrato inteligente cambió el estado de eDelivery a $Finished$ después de la ejecución de la función $finish()$. El protocolo asegura que cada receptor ha obtenido la clave de descifrado correcta desde que el contrato inteligente lo valida utilizando la ZKP (ver Subprotocolo 2 y Subprotocolo 3).

5. ***Confidencialidad*** La entrega será confidencial, solo los receptores que acepten la entrega pueden acceder a los datos entregados. Por esta razón, los datos no se pueden incluir en claro en ninguna transacción y no se deben registrar en la blockchain. Los datos no pueden ser un parámetro en las funciones del contrato inteligente y el contrato inteligente no puede obtener acceso a la clave de descifrado. Aún así, el protocolo debe verificar que los datos recibidos por todos los receptores sean los mismos. En el Subprotocolo 1, el remitente ejecuta la función de creación del contrato inteligente, incluidos C_1 y C_2, que representan el mensaje cifrado. En el Subprotocolo 2, cada receptor proporciona una manera para que el remitente envíe de forma privada la clave para descifrar el mensaje, a través de los elementos Z_1 y Z_2. El contrato inteligente verifica en el Subprotocolo 3 que se proporciona la clave de descifrado correcta y que se puede derivar de v, aislando r_i, al conocerse la clave privada b_i i el reto c_i. No hay más entidades involucradas en el intercambio, y las transacciones solo incluyen mensajes cifrados y claves de descifrado ocultas que solo son recuperables por los receptores que han aceptado el intercambio. Por lo tanto, los datos entregados serán confidenciales.

V. Conclusiones y trabajo futuro

Este trabajo propone un nuevo protocolo que logra el cumplimiento de todas las propiedades deseadas de un servicio de entregas certificadas utilizando blockchain. Hasta el momento estas propiedades se consiguieron parcialmente en los protocolos existentes. El nuevo esquema incluye la mejor parte de cada uno de los protocolos anteriores en un solo protocolo. De esta manera, el nuevo protocolo permite la entrega electrónica de datos confidenciales sin la necesidad de una tercera parte de confianza. Del diseño del nuevo protocolo se extraen las siguientes conclusiones:

- El protocolo presentado utiliza blockchain para permitir un intercambio equitativo para entregas certificadas. De esta forma, los datos, la prueba de origen y la prueba de recepción se intercambian sin la participación de ninguna TTP al tiempo que se logran las propiedades deseadas para los protocolos de intercambio equitativo: eficiencia, equidad, transferibilidad, no repudio y confidencialidad. Por esta razón, el protocolo es una mejora obvia de las propuestas anteriores.
- El protocolo permite entregas multiparte, útiles para la reducción de costes en aquellos escenarios en los que la misma entrega involucra a diversos receptores, como notificaciones a empleadeos, convocatorias a reuniones de accionistas,...
- Los datos entregados son confidenciales, solo el remitente y el receptor pueden acceder a los datos. Esto significa que el contrato inteligente debe gestionar el intercambio y garantizar la equidad sin poder acceder a los datos entregados. En términos de protocolo, esto significa que los datos entregados deben estar encriptados hasta la aceptación, cuando se proporciona la prueba de no rechazo de recepción, y el contrato inteligente, sin tener acceso a la clave de descifrado, debe garantizar que todos los receptores puedan descifrar el mismo mensaje.
- Para lograr todas las propiedades deseadas, incluida la confidencialidad, el protocolo utiliza una criptografía más compleja que los protocolos anteriores. El protocolo se ha implementado y probado en Ethereum para analizar el rendimiento de estas operaciones. Los resultados muestran que, aunque los costos de ejecución son ligeramente mayores que los de los protocolos no confidenciales, el protocolo es viable. Además, el uso de criptografia de curva elíptica permite utilizar longitudes de parámetros menores que os utilizados en protocolos garantizando el mismo nivel de seguridad. Esto permite controlar los costes asociados a la ejecución de los smart contracts.

Como trabajos adicionales vamos a realizar un estudio del volumen de datos que generaría el sistema, cuál sería la tasa de transferencia, y el coste económico del protocolo. Además, estudiaremos cuál sería el impacto si el cifrado planteado deja de ser seguro, ya que la información almacenada en una blockchain es inmutable y va a permanecer accesible de forma indefinida.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economia, Industria y Competitividad (MINECO), la Agencia Estatal de Investigacion (AEI) y European Regional Development Funds (ERDF) bajo el proyecto FeltiCHAIN RTI2018-097763-B-I00 (MINECO/AEI/ERDF, EU).

Agradecemos al Profesor Juan Tena su colaboración en el uso de los fundamentos de las Curvas elípticas aplicados en este artículo.

Referencias

[1] Hao, F.: "Schnorr Non-interactive Zero-Knowledge Proof". RFC 8235, September 2017. https://tools.ietf.org/html/rfc8235

[2] ISO/IEC 18033-2, Information Technology – Security Techniques – Encryption Algorithms – Part 2: Asymmetric Ciphers, International Organization for Standardization / International Electrotechnical Commission, 2006.

[3] V. Gayoso Martínez, F. Hernández Álvarez, L. Hernández Encinas and C. Sánchez Ávila, "A comparison of the standardized versions of ECIES, ", 2010 Sixth International Conference on Information Assurance and Security, Atlanta, GA, 2010, pp. 1-4.

[4] Q. Huang, G. Yang, D. Wong, and W. Susilo, "A new efficient optimistic fair exchange protocol without random oracles", International Journal of Information Security, 2012. Vol. 11, Num. 1, pp.53-63. ISSN 1615-5270, 2012.

[5] Q. Huang, D. Wong and W. Susilo, "P2OFE: Privacy-Preserving Optimistic Fair Exchange of Digital Signatures", Topics in Cryptology – CT-RSA 2014, Springer International Publishing. pp. 367-384, 2014.

[6] M. Payeras-Capellà, M. Mut-Puigserver and M. A. Cabot-Nadal, "Blockchain-Based System for Multiparty Electronic Registered Delivery Services", in IEEE Access, vol. 7, pp. 95825-95843, 2019. doi: 10.1109/ACCESS.2019.2929101

[7] N. Asokan, V. Shoup and M. Waidner, "Asynchronous Protocols for Optimistic Fair Exchange", Proceedings of the IEEE Symposium on Research in Security and Privacy, pp 86-99, Oakland, CA, May 1998.

[8] ETSI: ETSI EN 319 532-5 Electronic Signatures and Infrastructures (ESI); Registered Electronic Mail (REM) Services.
[9] European Union Agency for Network and Information Security: "Security Guidelines on the appropriate use of qualified electronic registered delivery services. Guidance for users, version 2.0, final, december 2016.
[10] H. Hasan and K. Salah, "Proof of Delivery of Digital Assets Using Blockchain and Smart Contracts,ÏEEE Access, vol. 6, pp. 65439-65448, 2018.
[11] J. Liu, W. Li, G. Karame and N. Asokan, "Towards Fairness of Cryptocurrency Payments", IEEE Security and Privacy, 2017.
[12] M. Mut-Puigserver, M. Payeras-Capellà and M. Cabot-Nadal, "Blockchain-Based Fair Certified Notifications", Data Privacy Management, Cryptocurrencies and Blockchain Technology", LNCS 11025, Springer International Publishing, pp = 20-37, 2018.
[13] M. Mut-Puigserver, M. Payeras-Capellà, M. Cabot-Nadal, "Blockchain-based Contract Signing Protocol for Confidential Contracts,"Proceedings of the IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), IEEE eXpress Conference Publishing, ISBN: 978-1-7281-5052-9, ISSN: 2161-5330, November 2019.
[14] J. A. Onieva, J. Zhou and J. Lopez, "Multiparty Nonrepudiation: A Survey". journal ACM Comput. Surv., Vol. 41, Num. 1, ISSN 0360-0300, pp. 5:1-5:43, ACM, New York, NY, USA, January 2009.
[15] M. Payeras-Capellà, M. Mut-Puigserver, M. Cabot-Nadal, "Smart Contract for Multiparty Fair Certified Notifications", Sixth International Symposium on Computing and Networking, Takayama, Japan, pp. 459-465, 2018.
[16] Z. Shao, "Fair exchange protocol of Schnorr signatures with semi-trusted adjudicator". Computers & Electrical Engineering, 2010.
[17] The European Parliament and the Council of the Europena Union: Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing. Directive 1999/93/EC.