IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 21, 2019, accepted July 5, 2019, date of publication July 16, 2019, date of current version August 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929101

Blockchain-Based System for Multiparty Electronic Registered Delivery Services

M. MAGDALENA PAYERAS-CAPELLÀ, MACIÀ MUT-PUIGSERVER, AND MIQUEL A. CABOT-NADAL

Departamento de Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, E-07122 Palma, Spain

Corresponding author: M. Magdalena Payeras-Capellà (mpayeras@uib.cat)

This work was supported in part by the Ministerio de Economia, Industria y Competitividad (MINECO), in part by the Agencia Estatal de Investigacion (AEI), and in part by the European Regional Development Funds (ERDF) through the FeltiCHAIN Project (MINECO/AEI/ERDF, EU) under Grant RTI2018-097763-B-I00.

ABSTRACT European Regulation (EU) 910/2014 lays down the rules for electronic identification and trust services for electronic transactions. Qualified Electronic Registered Delivery is one of the trust services included in the regulation, and it requires nonrepudiation of origin and reception together with the integrity of the data. This kind of services usually relies heavily on the use of trusted third parties. These entities are an obstacle to extend the use of protocols. In this paper, we use the blockchain-based technologies to provide, for the first time, two multiparty registered eDelivery schemes that reduce the involvement of trusted third parties compared with traditional approaches while satisfying the requirements of the norms created by the European Union for registered eDeliveries. Since confidentiality is not considered a compulsory property in the directive, we propose two protocols. The first one is well suited for those deliveries that do not require the confidentiality of the message or delivered data or even for those in which it is required that the message can be public and accessible to everybody. The second solution for multiparty registered eDelivery allows the message to be hidden to others than the receiver. We present the smart contracts for both proposals and also a complete analysis of their properties and performance.

INDEX TERMS Blockchain, registered electronic delivery services, certified notifications, smart contract, confidentiality, fairness, cryptocurrencies, certified electronic mail.

I. INTRODUCTION

Blockchain technology provides an unalterable system of data registry that enables new solutions for a wide range of traditional applications. Examples of these traditional services that could benefit from the distinctive features of blockchain are the provision of registered electronic delivery services, certified notifications or certified electronic mail, that is, a service that allows a sender to prove that she has sent a message or other kind of data, as a file, to a receiver or set of receivers. Thus, these services provide evidence that a receiver has access to the information since a specific date/time. Generally speaking, trust services must be secure, protect the privacy of the users and at the same time they have to take into account the existing regulations.

Registered delivery services, along with other electronic services, such as electronic signature of contracts or electronic purchases (payment in exchange for a receipt or digital product), require a fair exchange of items between two or more users. In order to create protocols that allow carrying out these exchanges and, at the same time, maintain the security of the communications, there are solutions that fall into the generic pattern named fair exchange of values. A fair exchange always provides an equal treatment to all users, and, at the end of each execution, either each party has the element it wishes to obtain from the other party, or the exchange has not been carried out successfully for anyone (any party has not received the expected item).

To solve the problem of fair exchanges, traditional solutions include trusted third parties (TTPs by Trusted Third Parties) managing the exchanges to a greater or lesser extent. In a typical registered eDelivery case, the element to be exchanged is the delivered data along with non-repudiation proofs of origin and reception.

This paper aims to show how the blockchain technology and the smart contracts can introduce a new paradigm to deal

The associate editor coordinating the review of this manuscript and approving it for publication was Kim-Kwang Raymond Choo.

with the fair exchange problem. By using this technology, we can reduce or even remove the role of the TTPs inside such protocols.

A. QUALIFIED ELECTRONIC REGISTERED DELIVERY SERVICES

On July 2016, the Regulation (EU)910/2014 of the European Union Agency for Network an Information Security [30],also known as eIDAS Regulation, became applicable. This regulation establishes the rules on electronic identification and trust services for electronic transactions in the internal market and covers all 28 member states.

Several trust services are defined in the document: electronic signatures, seals, timestamps, registered delivered services and certificates for website authentication. The regulation introduces the notions of qualified trust service and qualified trust service provider with the requirements and obligations that ensure high-level security of the trust service. Also, there are a series of documents, like [7] that aim to assist parties wishing to use the aforementioned services.

In the US we can find the FICAM Trust Framework Solutions (TFS),[1] the federated identity framework for the U.S. federal government. It includes guidance, processes and supporting infrastructure to enable secure and streamlined citizen and business facing online service delivery. However, the framework does not include the specifications for the registered electronic delivery of data.

We will focus on the electronic registered delivery service, also called *eDelivery*. For simplicity, we will use the term eDelivery in the proposed protocols. Data sent and received using this service achieve the properties of integrity of the data, the identification of the sender of the data and also of its receiver. Examples of use of the qualified electronic registered delivery services are: electronic registered mail, official submissions in e-government services, access and exchange of sensitive data and notarization of events. It is important to highlight that the eIDAS regulation establishes the principle that an electronic document should not be denied legal effect because it is in electronic form.

Article 3.36 of the regulation [30] states that an eDelivery service is ''a service that makes it possible to transmit data between third parties by electronic means and provides evidence related to the handling of the transmitted data, including proof of sending and receiving data, and that protects transmitted data against the risk of loss, theft, damage or any unauthorized alterations''.

A qualified eDelivery service must offer the following functionalities according to the directive: data integrity, authentication of origin (both natural person or legal person) and authentication of the time of the delivery. Confidentiality is not considered a core functionality but it is usually provided as part of a more complete solution. This consideration will be taken into account in the design of the protocols. For an eDelivery service in order to be a Qualified Electronic Registered Delivery Service according to [30], the service must be provided by qualified trust service providers whose compliance with the requirements is regularly checked by an accredited entity.

The data, that can be sent in an eDelivery service from a sender to a receiver, can be of any type (thus, it includes electronic documents) and the transmission means can be of any kind. Usually the data referred to by the definition of eDelivery services is generally called a 'message'. This way, certified notifications and certified electronic mail are included in the eDeliviery services. eMail is a transmission mean that can be used but eDelivery is not limited to eMail. Registered eMail is a general-purpose implementation of eDelivery [7], whereas there are more specific eDelivery services.

Some use cases for eDelivery services are included in [7]:

- Electronic registered mail, an enhanced form of email transmitted by electronic means that provides evidence relating to the handling of an e-mail including proof of submission and delivery.
- Delivery of official electronic notifications and supporting official submissions in eGovernment services.
- Accessing and exchanging sensitive electronic data.
- Electronic notarization of events.

In addition to that, we now introduce the notion of multiparty eDelivery system. A multiparty fair exchange scenario can be a special case of eDelivery. In such scenario different parties are involved in an exchange of messages. Reference [25] defines a multiparty non-repudiation scheme as a protocol where ''$n \mid n > 2$ entities agree to use a non-repudiation protocol for exchanging messages (general or specific purpose) and collecting evidence of the transactions performed for the exchange of those messages''. Of course, the way of exchanging the message among the different actors of the protocol can be different depending on the specific application. For example, the actors can be distributed as a ring, a mesh or the distribution of the message can be one-to-many.

B. FAIRNESS AND TTPs IN EXCHANGE PROTOCOLS

In order to achieve fairness, the protocols proposed so far for operations including exchange of elements (e.g. eDelivery services) usually use TTPs, which are responsible for resolving any conflict that arises as a result of interrupted exchanges or fraud attempts. In addition to that, these protocols normally use non-repudiation mechanisms in order to generate evidence that proves the behavior of the actors of the protocol, so that, in case of dispute, an external arbitrator can evaluate them and make an unambiguous decision to guarantee the fairness of the exchange. Then, current fair exchange protocols [16], [17], [28] include the intervention of TTPs that have similar functions to what we could call an electronic notary or a judge.

However, in practice, the implementation and acceptance of this type of entities is an obstacle to extend the use of protocols in the network. On the one hand, it is difficult

[1] https://www.idmanagement.gov/trust-services/

to have TTPs that are really reliable for any user in the network and that have a defined framework of action (e.g. the electronic documents generated by the TTP have to be accepted to resolve disputes in a court of law in different countries). On the other hand, TTPs can also cause problems at a technical level (e.g., they can cause bottlenecks from the point of view of communications), lack of efficiency in the protocols (e.g., slow down the resolution of problems) and increase the cost of the execution of the protocol (e.g. charging high rates for the provision of services). Besides, they are a very sensitive point in the network since they play an important role in the security of electronic protocols and their reliability is a problem that needs attention, because the security of the exchange can be broken if the TTP has any vulnerability.

C. SOLUTION ON THE BLOCKCHAIN

Although TTPs are still a basic actor in fair exchange protocols, currently, with the advent of the blockchain technology and smart contracts, TTPs could be replaced or complemented by this new know-how, which opens a range of new possibilities to find effective solutions to the electronic versions of the protocols that fulfill the generic pattern of fair exchange of values.

In addition to that, smart contracts reduce the need for trusted intermediaries or reputation systems. This means that this new technology allows us to define transactions with predetermined rules (written in a contract) in a programmable logic that can guarantee a fair exchange between parties with an initial mutual distrust. This feature prevents parties from cheating each other and discharges the need for intermediaries with the consequent reduction of delays and commissions for their services.

It is expected that such systems will stimulate new forms of fair exchanges. The revealing power of the blockchain is further enhanced by the fact that blockchains naturally incorporate a discrete notion of time, a clock that increases each time a new block is added. The existence of a trusted clock/register is crucial to achieve the property of fairness in the protocols.

D. ORGANIZATION OF THE PAPER

The paper is organized as follows: after the introduction, the ideal properties of Registered eDelivery systems are listed in Section §II. The state of the art on the subject is reviewed in Section §III. The contribution of the paper is depicted in Section §IV. After this, we include the system overview (§V) and the two proposals for the blockchain-based multiparty eDelivery; Section §VI presents the protocol and the development of the non-confidential blockchain-based multiparty registered eDelivery protocol while Section §VII does the same with the confidential multiparty protocol. Section §VIII presents the implementation and the Smart Contracts for both protocols. Then the analysis of the system is performed; Section §IX includes the analysis of properties while Section §X analyses the performance of the system in terms of cost and delay. The conclusions are listed in Section §XI.

II. IDEAL PROPERTIES OF A REGISTERED eDELIVERY SYSTEM

The document [30] includes a list of the key features of a registered eDelivery service, divided into legal features, security features, functional features and other. These requirements are related with the integrity of the data, the sending of the data by an identified sender, its receipt by the identified addressee and the accuracy of the date and time of sending and the date and time of receipt.

- **Legal Features.**
  - *Proof of Delivery.* The sender receives an unforgeable time stamped proof of the instant she has delivered a given message to a receiver.
  - *Proof of Reception.* Both the sender and the recipient receive an unforgeable time stamped proof of the instant the receiver received or opened the electronic message
  - *Proof of Integrity.* Both the sender and the receiver can be sure that the message has not been changed during transmission.
  - *Protection.* Protection against loss, theft, damage or unauthorized alterations.
- **Security Features.**
  - *Sender Identification.* The receiver of the message can authenticate the sender of the received message.
  - *Secure Time Stamping.* All time indications are protected by a qualified electronic timestamp. This timestamp provides legal presumption of the accuracy of the date and time of the delivery
  - *Confidentiality.* Both the sender and the receiver can be sure that the message cannot be accessed by unauthorised persons.
  - *Data Integrity.* This property ensures the integrity of the transmitted data, that is, the content of the message.
  - *Control on routing errors.* This control helps the user to check different parameters of the receiver before the transmission and inform the user about the ability of the receiver to accept the message before the transmission.
  - *Interoperability.* A service indicates to the sender all formats of messages that the intended receiver can process and transform a message from one format to another.
- **Functional Features**
  - *Sending of large files.* The service must allow large messages and messages of all kinds of formats to be transmitted.
  - *Fast Processing.* The delivery must be instantaneous.

- **Other Properties**
  - *Reduced Risk.* Qualified electronic registered delivery makes it infeasible to manipulate data, to forge timestamps of sending and receiving, or to provide unauthorised access to the message. These services are provided by trust service providers.
  - *Reduced Cost.* The service avoids logistical costs and reduces the cost of transmission failure or uncertainty.
  - *No Transmission Delays.* Transmission has to be virtually instantaneous.
  - *No Double Sending.* Avoids the sending of an additional signed version, in paper, of the electronic data.
  - *Incident Handling and Liabilities.* The service provider remains liable for damage that was made to its customer by negligence or omission.

The above features must be taken into consideration in order to make a list of the security properties that should fulfill an eDelivery system. In addition to that, the requirements for fair exchange were stated in [1] and re-formulated in [31]. Consequently, we have grouped and summarized such properties, bearing in mind that the eDelivery service is a special case of a fair exchange of values. The Ideal Properties of a Registered eDelivery System are:

1) **Effectiveness.** If two parties behave correctly, they will receive the expected items.
2) **Fairness.** After completion of a protocol run, either each party receives the expected item or neither party receives any useful information about the other's item. The fairness is *weak* if, by the end of the execution, both parties have received the expected items or if one entity receives the expected item and another entity does not, the latter can get evidence of this situation.
3) **Timeliness.** At any time, during a protocol run, each party can unilaterally choose to terminate the protocol without losing fairness. In addition to that, a protocol can achieve *Weak Timeliness* if any honest party can be sure that any protocol run will be concluded at a certain finite point in time. That is, the state of the exchange will be final from the party's perspective at this completion point.
4) **Non-repudiation.** If an item has been sent from party A to party B, A cannot deny origin of the item and B cannot deny receipt of the item.
5) **Verifiability of Third Party.** If the third party misbehaves, resulting in the loss of fairness for a party, the victim can prove this fact in a dispute.
6) **Confidentiality**. Only the sender and the receiver of the data know the contents of the certified message.
7) **Efficiency.** An efficient protocol uses the minimum number of steps that allow the effective exchange or the minimum cost.
8) **Transferability of evidence.** The proofs generated by the system can be transferred to external entities to prove the result of the exchange.
9) **State Storage.** If the TTP that can be involved in the exchange is not required to maintain state information then the system is *stateless*.

In addition to that, we have to make some remarks to highlight the link between the above properties and the features described in [30]:

- Legal and security features are closely related to the fairness and non-repudiation security properties. Onieva et al. explained the non-repudiation phases in [25]: *Evidence Generation*, *Evidence Transfer*, *Evidence Verification*, *Evidence Storage* and *Dispute Resolution*. The generation of proofs or evidence is the previous step to the provision of non-repudiation services and the ability to demonstrate the fairness of an exchange in case of dispute resolution. Obviously, proper timestamp mechanisms and data integrity checks have to be established in order to make an appropriate evaluation and verification of the generated evidence.
- Some of the above properties cannot be achieved in the same protocol. The authors of [9] enumerate the incompatibilities among the ideal features. Some examples are Weak Fairness versus Transferability of evidence, Stateless TTP versus Timeliness and Verifiability of the TTP versus Transparency.

III. STATE OF THE ART OF REGISTERED EDELIVERY SERVICES

eDelivery follows the pattern of fair exchange of values. This kind of exchange does not have a definitive and standardized solution in its electronic version. The notification of a message can be done using electronic mail and, until now, several proposals have been presented for this service. However, it is not required that the eDelivery use electronic mail. It includes an exchange of elements between the sender and the receiver; the sender has to send a message to the receiver, then the receiver is able to read it and, in exchange, the receiver has to send a proof of reception to the sender.

To overcome reluctance between the parties and to assure fairness, almost all the existing proposals use a TTP. This trusted third party can play and important role, participating in each exchange, or a more relaxed role in which the TTP is only active in case a dispute arises between the parties (optimistic protocols). Due to the incompatibility among some of the properties and the difficulty to achieve simultaneously some other properties, we can find protocols that solve the exchange in an efficient way with an optimistic TTP although achieving only weak fairness [9], other systems focused in the achievement of specific features as the transferability of evidence [20], the verifiability of the TTP [22], the avoidance of the selective rejection based on the identity of the sender [26], the flexibility to allow the delivery to multiple receivers [10] or the reduction [24] of the volume of state information that the TTP must maintain.

Previous studies [2], [8], [15] on fairness using blockchain focus on fair purchase operations between a product (or a receipt) in exchange for cryptocurrencies (usually bitcoin).

Reference [21] uses, for the first time, a smart contract for the resolution of a purchase operation. The authors investigate the fair exchange problem and propose two solutions for fair payment using smart contracts supported by the functionality of blockchain with a Turing-complete language. Thus, most of the solutions using the blockchain technology for fair exchanges are focused in performing atomic payments. For example, in [5] Delgado-Segura et al. propose a protocol based on Bitcoin scripting language for a fair exchange between data and a payment where the seller of the data cannot cash in the payment if the buyer has not obtained the data and the buyer will not get the data without executing the payment. In [12] some variants of a protocol for exchanging Bitcoins for digital goods are presented by using an escrow system. Also in [13] another problem related to fair exchange and the blockchain technology is studied. The authors propose a solution for a Proof of Delivery of physical assets without the involvement of any TTP. The provided solution can be applied to many shipped physical items in exchange for a payment. In [14], Hasan et al. propose a non-repudiation protocol using the blockchain technology and the Ethereum smart contracts. This solution is applied to a trade exchange between crypto-tokens and digital assets. The protocol requires a deposit of a collateral by the involved parties in order to incentivize the honest behavior of the actors. Thus, this protocol is only appropriate for trade scenarios to provide proof of delivery but the fairness of the exchange between sender and receiver is not proven.

We have recently published [23], [27] two works dealing with the incorporation of blockchain in certified notification protocols. As far as we know, there are no other works that deal with blockchain-based registered delivery services and smart contracts. Reference [23] has two proposals, the first one enables a non-confidential fair exchange of a notification message for a non-repudiation of reception token with no involvement of any TTP. The second one allows a confidential fair exchange of a notification for a non-repudiation of reception token. It has the optimistic intervention of a stateless TTP. In addition to that, in [27] we introduced the notion of reusable smart contracts for sending several notifications.

A multiparty eDelivery allows the sender to send a message efficiently to multiple receivers. Usually such protocols make use of TTPs in optimistic approaches to solve the exchange and don't use blockchain. Reference [19] presents a protocol that uses group signatures with an on-line TTP, [10] presents an efficient multiparty optimistic protocol with the properties of asynchrony and verificability of the TTP while [32] describes a very efficient, asynchronous optimistic protocol. Also, in [18] there is a proposal to solve a multiparty fair exchange that it can work asymptotically optimal for any topology (it needs a constant number of rounds) and it completely relies on the TTP to achieve fairness. However, as far as we know, this paper is the only proposal that exists that use blockchain to deal with the problem of multiparty fair exchanges.

IV. CONTRIBUTION

In this paper we will analyse the possibilities of the use of blockchain-based technologies to provide, for the first time, two multiparty registered eDelivery schemes that reduce the involvement of trusted third parties compared with traditional approaches while satisfying the requirements of the normative created by the European Union for registered eDeliveries. Since confidentiality is not considered a compulsory property in the directive [22] we propose two protocols. The first proposed protocol is well suited for those deliveries that do not require the confidentiality of the message or delivered data, or even it is required that the message can be public and accessible to everybody. Then, a second solution for multiparty registered eDelivery allows the message to be hidden to others than the receiver.

The multiparty registered eDelivery protocols are far more efficient than two-party protocols. Moreover, the use of blockchain-based technologies allows the definition of more attractive solutions, reducing its dependence on third parties. In the first approach, there is no need of a TTP in any step of the exchange nor in a dispute resolution phase while, in the second proposal, the trusted third party will be involved only in a dispute resolution phase in case it is necessary (optimistic protocol). Moreover, it is not required that this TTP stores information of the state of any transaction.

In addition to that, in this paper we also present the corresponding smart contracts to implement blockchain-based services for the proposed protocols. The first one manages non-confidential multiparty eDeliveries and the second one confidential multiparty eDeliveries. The differences in the protocols cause an important contrast in the design and the implementation (who creates the smart contract, when, who pays for the service,...). We will justify these criteria in the description of the implementation of the protocols.

Moreover, the protocols are fully analyzed. We have performed an analysis of properties to prove that the proposals satisfy the requirements listed in the legislation and also the ideal properties for this kind of protocols (i.e. fair exchange protocols). This analysis includes six propositions and fourteen claims. The second analysis is a performance analysis where the cost to deploy the smart contract and the cost to execute the main functions are compared and discussed. The delays caused by these functions are also studied.

The contribution of the paper can be summarized as the first proposal for multiparty registered e-Delivery based on blockchain satisfying the fulfillement of properties, privacy and performance requirements.

In our organization account at GitHub, *SECOM Research Group*, at https://github.com/secomuib, the proposals can be found in the following repositories:

- *NonConfidentialMultipartyRegisteredEDelivery*[2]
- *ConfidentialMultipartyRegisteredEDelivery*[3]

[2]https://github.com/secomuib/NonConfidentialMultipartyRegisteredEDelivery
[3]https://github.com/secomuib/ConfidentialMultipartyRegisteredEDelivery

V. SYSTEM OVERVIEW

This section presents an overview of the system, describing the participating actors, the two proposed alternatives, the methods and the interactions among the actors.

The proposed system presents solutions for both confidential and non-confidential (or public) certified *eDelivery* of messages. The proposals consider the following actors with these roles:

- Sender (A). User that generates the data to deliver, chooses the receiver or set of receivers and sends the message. The sender also provides a non-repudiation of origin proof.
- Receivers (B). User or set of users that receive the delivery. The receivers must accept the delivery and provide a non-repudiation of reception proof. The delivery can be accepted by a subset of receivers so the protocol must manage the exchange to maintain fairness in case of a selective acceptance.
- Trusted Third Party (*TTP*). Independent and trusted party that can act as an intermediary to solve disputes among senders and receivers. Only the confidential protocol involves a *TTP*.
- Smart Contract. Contract deployed on the blockchain that can manage, depending on the proposal, both the exchange of elements during the delivery and/or the resolution of disputes among senders and receivers.

All the participating actors (senders, receivers and TTP when it is required), possess blockchain addresses and are able to communicate with each other and with the smart contract. These entities interact as follows: first the sender chooses whether he wants to send a confidential notification or a public notification. Then, in both protocols, the parties execute a three-step exchange to deliver the message and provide non-repudiation proofs:

- In the first step the sender proposes the message to be delivered and sends it in a hidden way to the set of receivers. This step must include elements to assure the fairness of the exchange.
- Each receiver can choose individually if he accepts or not the delivery. The receivers who accept the delivery must send a non-repudiation of reception proof related with the hidden message.
- The sender determines the group of receivers that has accepted the delivery and concludes the exchange providing the message in clear and the non-repudiation of origin proof.

Both proposed protocols follow the three step exchange. The main difference between the non-confidential and the confidential solution in regard to this three-step protocol is that in the confidential solution the parties can exchange messages directly (*off-chain* communication exchange), while in the non-confidential solution the parties execute the three-step protocol *on-chain*, by invoking functions of the smart contract deployed for this service. The *on-chain* interaction among the actors for the non-confidential protocol is depicted in Figure 1, while the *off-chain* interaction for the confidential protocol is depicted in Figure 2.

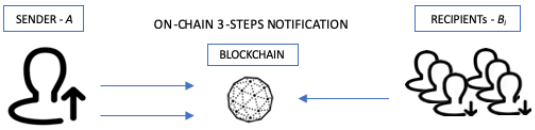

FIGURE 1. Interaction among the actors in the non-confidential protocol.

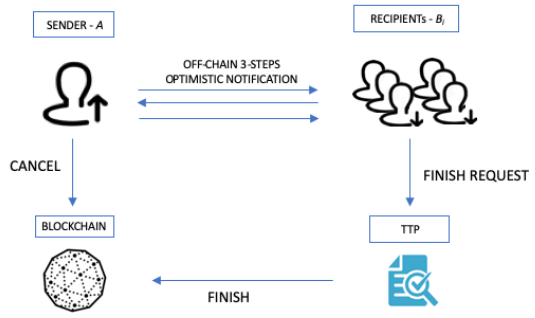

FIGURE 2. Interaction among the actors in the confidential protocol.

Due to the differences in the execution of the three-step protocol, a conflict resolution subprotocol has to be designed for the confidential solution in order to assure fairness. This subprotocol uses the smart contract for the confidential *eDelivery* service and involves a trusted third party, as it is depicted in Figure 2.

Moreover, the three steps of the exchange are slightly different in the two protocols. They differ in the way they hide the data to deliver and also in the format of the non-repudiation proofs, as it will be explained in sections VI and VII.

VI. NON-CONFIDENTIAL BLOCKCHAIN-BASED MULTIPARTY REGISTERED eDELIVERY PROTOCOL

In [23] we presented a blockchain-based protocol for Certified Notifications for a single receiver. The proposal is a non-confidential fair exchange of a notification message together with a non-repudiation of origin token for a non-repudiation of reception token. This approach is well suited for those applications that require the storage of the delivered data, that must be registered and publicly accessible. However, the scheme is a two-party protocol, thus the notification can only have a single recipient.

Now, in this paper, we present an improved non-confidential protocol with the new property of reusability and multiparty capabilities (i.e. multiple receivers), for registered *eDeliveries*.

The multiparty protocol for non-confidential registered *eDeliveries* presents a solution for fair deliveries with one Sender (A) and multiple Recipients ($B = \{B_1, B_2..., B_n\}$). Figure 3 is the diagram of the sequence of the messages exchanged by Sender and Recipients in the non-confidential *eDelivery* for the on-chain communication scheme proposed. In Figure 3 the blue arrows represent the steps described above and designate the signed requests to a

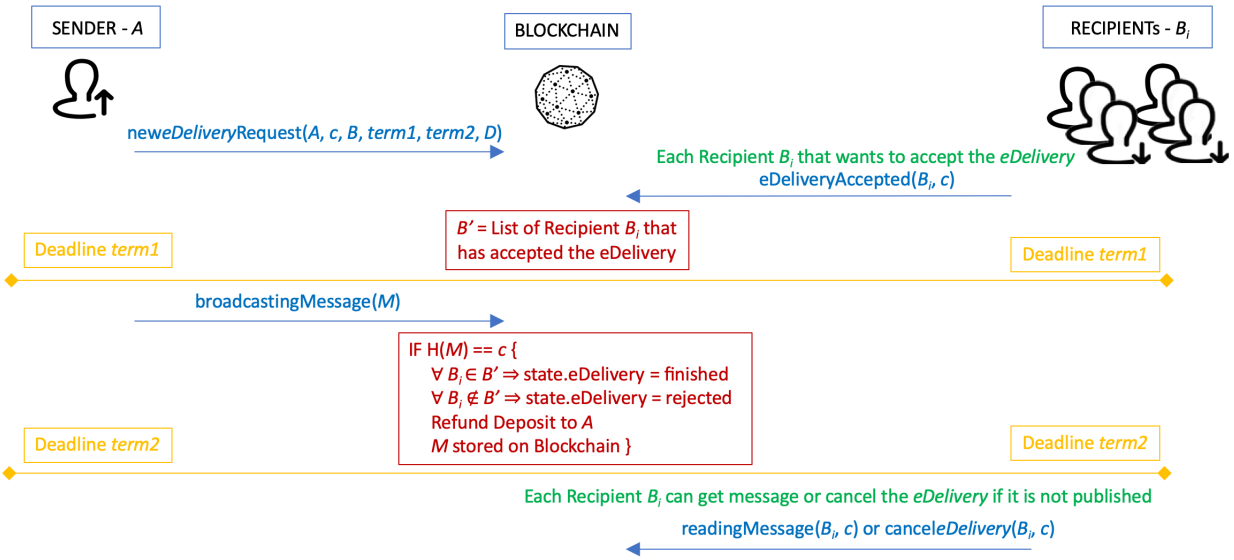

FIGURE 3. Non-confidential blockchain-based protocol description.

TABLE 1. Elements of the non-confidential multiparty protocol.

Elements	
A	Blockchain address of the sender A
B	Set of receivers
M	Message, data or file to deliver
B_i	Blockchain address of the receiver B_i
B'	Subset of B with the users that have accepted the delivery.
$term1$	Period for receivers to accept the delivery, specified as time since its creation.
$term2$	Period to allow receivers to cancel the delivery, specified as time since its creation.
$c = H(M)$	Collision-resistant hash function applied to M.
D	Deposit sent to the *eDelivery* Service.

blockchain address. Also, the text in red inside the boxes describes the processes that have to be performed by the *eDelivery* service deployed on the blockchain.

The sender of the delivery, A, and the set of receivers, B, will follow the steps of the exchange protocol depicted in Section V. In the following complete description of the protocol, the requests sent by the actors of the protocol are directed to the address of the *eDelivery* service deployed on the blockchain. The details of the exchange protocol are (see Table 1 for notation):

1) The sender A sends a request to create a new *eDelivery*. This request includes the identification address of the sender, the hash of the message to deliver (c, this hash code is also used as identification number of the *eDelivery*), the addresses of the receivers and the periods $term1$ and $term2$. $term1$ specifies the valid period for the receivers to accept the delivery before the sender finishes the delivery, and $term2$ specifies the time to allow receivers to cancel an unfinished delivery. We will see in Section IX that this cancellation is required to guarantee effectiveness and fairness. A deposit is used in this transaction to encourage finalization, but it could be optional.
2) Each receiver B_i in B has to individually accept the reception of the delivery during $term1$, publishing a message expressing his will. This signed transaction will act as a non-repudiation of reception proof. If a receiver does not accept during $term1$, then a rejection is assumed.
3) After the deadline of $term1$, or after all receivers (members of B) have accepted the reception, sender A can publish the message by using the blockchain, finishing the delivery with the subset of receivers B' ($B' \subset B$) that has accepted the delivery. As a consequence, the *eDelivery* service deployed on the blockchain checks the integrity of the message and publishes the non-repudiation of origin proof for the receivers in B'. After the finalization, the sender receives the refund of the deposit.
4) In this case, we have add a final step to the general three-step protocol: after the deadline of $term2$ any receiver in B' can get the message or can request the cancellation of the *eDelivery* in case the message was not properly deposited in the previous step.

Finally, after the execution of the exchange protocol:

- All the receivers can read the message M since it is stored in the blockchain, but only members of B' can prove that they have been notified and have non-repudiation of origin proofs.
- If after $term2$ the sender A hasn't published the message, each receiver B_i can cancel the eDelivery. In this case, the state of this receivers will be *cancelled*.

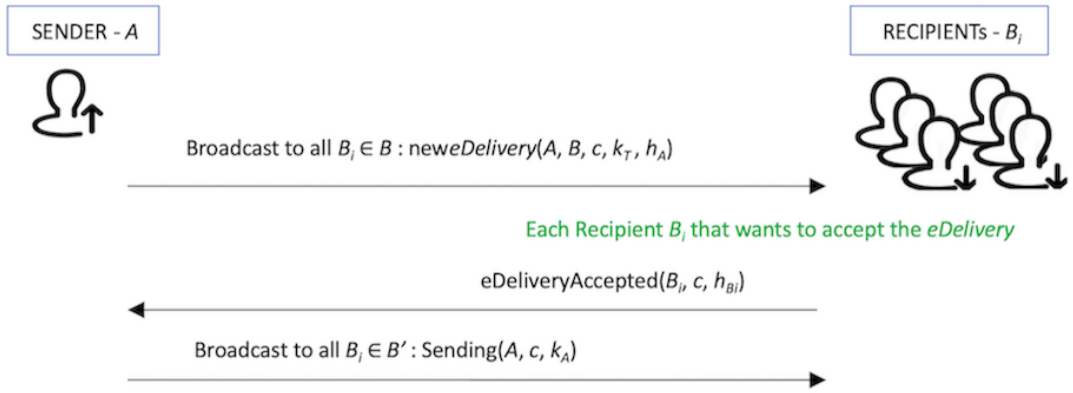

FIGURE 4. Optimistic off-chain communication exchange subprotocol .

VII. CONFIDENTIAL BLOCKCHAIN-BASED REGISTERED eDELIVERY PROTOCOL

The second proposal has been designed taking into account those deliveries that require confidentiality. That is, the blockchain has to help to preserve the fairness of the exchange but the message cannot be stored in a publicly accessible block.

The main difference with the non-confidential one is that the protocol, in the confidential case, allows an off-chain optimistic exchange, that is, the exchange can be executed completely without the intervention of the blockchain nor the *TTP*.

Another important feature is that this proposal does not require a deadline and any exchange can be finished at any moment. A stateless *TTP* can be used to resolve the disputes that could arise between the parties.

In [9], [10] we described a non-blockchain-based optimistic fair exchange that we partially reuse (i.e. the off-chain three-step exchange) and adapt for this purpose in the new blockchain-based proposal described in this section. In the new protocol, the originator A and the set of recipients B exchange messages and non-repudiation evidence directly, using the three-step off-chain communication depicted in Figure 4. Only as a last resort, in case they cannot get the expected items from the other party, the smart contract or the *TTP* would be invoked, by sending a *cancellation request* (Figure 5) or a *finish request* (Figure 6). In comparison with the protocol described in [10], the role of the *TTP* has been substantially reduced in the blockchain-based solution. Moreover, the sender will never contact the *TTP*. In this new proposal, the *TTP* will answer only requests from the receivers (members of B) by accessing the smart contract that has been deployed. The *TTP* is totally stateless, so it never stores information about the state of any exchange.

Our new multiparty protocol has three subprotocols. Next, we explain in detail these subprotocols, including the values that must be exchanged by the different parties. The subprotocols are: *Optimistic Exchange*, *Cancel* and *Finish* Subprotocols. Note that blue arrows in Figures represent signed request to a blockchain address, while black arrows are off-chain communication messages. Also, the text in red inside the boxes describes the processes that have to be performed by the *eDelivery* service deployed on the blockchain. Moreover, the description of the protocol follows the notation included in Table 2.

TABLE 2. Elements of the confidential multiparty protocol.

Elements	
M	Message, data or file to deliver.
A	Sender of the delivered data.
B	Set of Receivers.
B_i	A specific receiver.
B'	Subset of B with the users that will finish the exchange with A.
B''	Subset of B with the users that will not finish the optimistic exchange with A.
$B'' - finished$	Members of B that have contacted the TTP before the cancellation of the exchange.
$B'' - cancelled$	Members of B that have not contacted the TTP before the cancellation of the exchange.
$c = E_K(M)$	Symmetric cipher of message M with key K.
$k_T = PU_T(K)$	Key K encrypted with TTP's public key
$h_A = PR_A[H[H(c), B, k_T]]$	A's signature (using her private key) on the concatenation of the hash of c, B and k_T and Id. First part of Non-Repudiation of Origin proof.
$h_{Bi} = PR_{Bi}[H[H(c), k_T]]$	B_i's signature (using his private key) on the concatenation of the hash of c and k_T and Id. Non-Repudiation of reception proof.
$k_A = PR_A[K, B']$	A's signature (using her private key) on the key K together with B'. Second part of the Non-Repudiation of origin proof for $B_i \epsilon B'$.
$k'_T = PR_T[K, B_i]$	TTP's signature (using its private key) on key k for user B_i. Alternative second part of the Non-Repudiation of origin proof.
$h_{BiT} = PR_B[H[H(c), k_T, h_A, h_{Bi}]]$	Evidence of the TTP intervention requested by B_i.

A. MULTIPARTY OPTIMISTIC EXCHANGE SUBPROTOCOL

The protocol is optimistic in the sense that it is possible for a sender A to complete the exchange with the set of receivers B without the intervention of the *TTP*. Figure 4 depicts the three-step off-chain message sequence exchanged by sender and recipients of a confidential *eDelivery* to solve the exchange through the optimistic approach.

The exchange is as follows:

1) The sender sends a message to the set of receivers including the encrypted message, the addresses of the receivers and the first part of the non repudiation of origin proof, k_T, h_A.

2) Each receiver decides if he follows the exchange sending the non repudiation of reception proof, h_{Bi}.
3) The sender sends to each receiver, which has sent the message of step 2, a message containing the decryption key. Thus, this subset of receivers will receive the key to open the message and the remaining part of the non repudiation of origin proof, k_A.

If the execution of these steps has been successfully completed, the sender will hold non-repudiation of reception (NRR) evidence from all recipients and every recipient will hold the message and non-repudiation of origin (NRO) evidence. Every recipient has the key and so he can decrypt the message, then each of the recipients of the set B obtains the key used to decipher the message, k_A, as well as the corresponding NRO evidence (h_A). In the same way, the sender of the message will obtain the NRR evidence (h_{Bi}) from each recipient. If some of the recipients don't send the message of step 2, those recipients won't receive the message of the last step. In fact, this last message contains the list of recipients that have completed the protocol. Therefore, a receiver that is not in the subset B' cannot use the message of the step 3 received by other receivers as a NRO evidence. If some party does not follow this exchange protocol, the remaining users need to correct the unfair situation by requesting the cancel or finish resolutions.

This way, the protocol allows an optimistic exchange, that is, the exchange can be executed completely without the intervention of the *TTP* nor the blockchain. Another important feature is that this proposal does not require a deadline and can be finished at any moment. The following subprotocols can be executed if disputes arises between the parties.

B. MULTIPARTY CANCEL SUBPROTOCOL

The Cancel subprotocol will be initiated by the sender of the message. The sender executes the corresponding function of the smart contract in case of not receiving the element h_{Bi} from all the recipients of the messsage. In Figure 5 there is a graphical description of the actions taken by the sender and the blockchain to cancel a confidential *eDelivery* for non-finished recipients. The hash code $H(c)$ is used as identification number of the eDelivery.

When the sender executes the *Cancel* function of the smart contract, she has to indicate the identity of all those users who have not sent h_{Bi} (represented by the set B''). The smart contract, for its part, will be responsible for checking if any of the users in the set B'' has already finished the exchange by means of the *TTP*. In this case, it will send the corresponding NRR evidence (for a particular B_i) to the sender. Otherwise the unfinished recipients will be included in the group of cancelled users ($B'' - cancelled$). Therefore, at the end of this phase, the sender A will have concluded the fair exchange with all recipients, either satisfactorily, because she has received the correspondent h_{Bi}, or as a result of a cancellation.

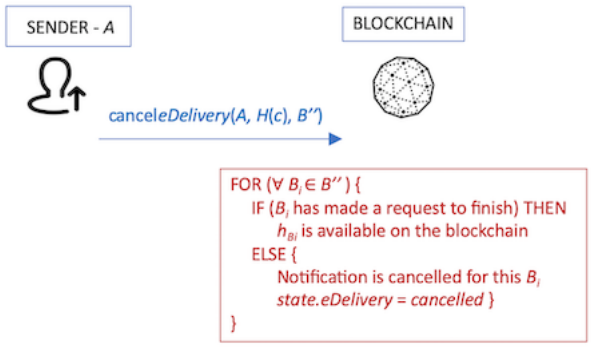

FIGURE 5. On-chain cancel subprotocol .

C. MULTIPARTY FINISH SUBPROTOCOL

The Finish subprotocol will be initiated by any receiver, in the case of having sent the corresponding h_{Bi} but not having received the element to obtain the encryption key, k_A. This finalization will be carried out by the *TTP* based on the request received from a recipient B_i. After checking the correctness of all the different parameters received from B_i the *TTP* executes the *finish* function of the smart contract (the *TTP* submits B_i's NRR evidence (h_{Bi}) to the smart contract). In Figure 6 there is the description of the actions taken by recipients, *TTP* and blockchain to finish a confidential *eDelivery* in case of exception. The *TTP*'s response is based on the information stored on the blockchain about this *eDelivery*.

In this subprotocol, the smart contract checks that the request comes from the *TTP*, then it verifies if the claimed recipient is among the users whose message delivery has been cancelled. In this case, the appropriate cancellation evidence is issued. Otherwise, the smart contract stores in the blockchain the received h_{Bi} and updates the set of users who have finished this exchange by adding B_i to $B'' - finished$. Finally, the *TTP* sends k_T to B_i in order to enable the recipient to read the message and to complete the NRO evidence.

VIII. SMART CONTRACTS

To implement the protocols depicted in Sections §VI and §VII, we have used the Ethereum blockchain, because it offers an even richer functionality set than conventional blockchains such as Bitcoin, since they support programs called *Smart Contracts* in a fully distributed system that stores the changes in the system's state. The smart contracts use a programming language that is *Turing complete*, which converts this platform as a general-purpose computer. In this blockchain, its cryptocurrency, *Ether*, is used to meter and constrain execution resource costs.

A. SMART CONTRACT FOR THE NON-CONFIDENTIAL MULTIPARTY REGISTERED eDELIVERY PROTOCOL

In this protocol, we have used a *Smart Contract* to allow the sender A to send the same non-confidential message to several receivers, members of B, exchanging non-repudiation evidence to complete the *eDelivery*.

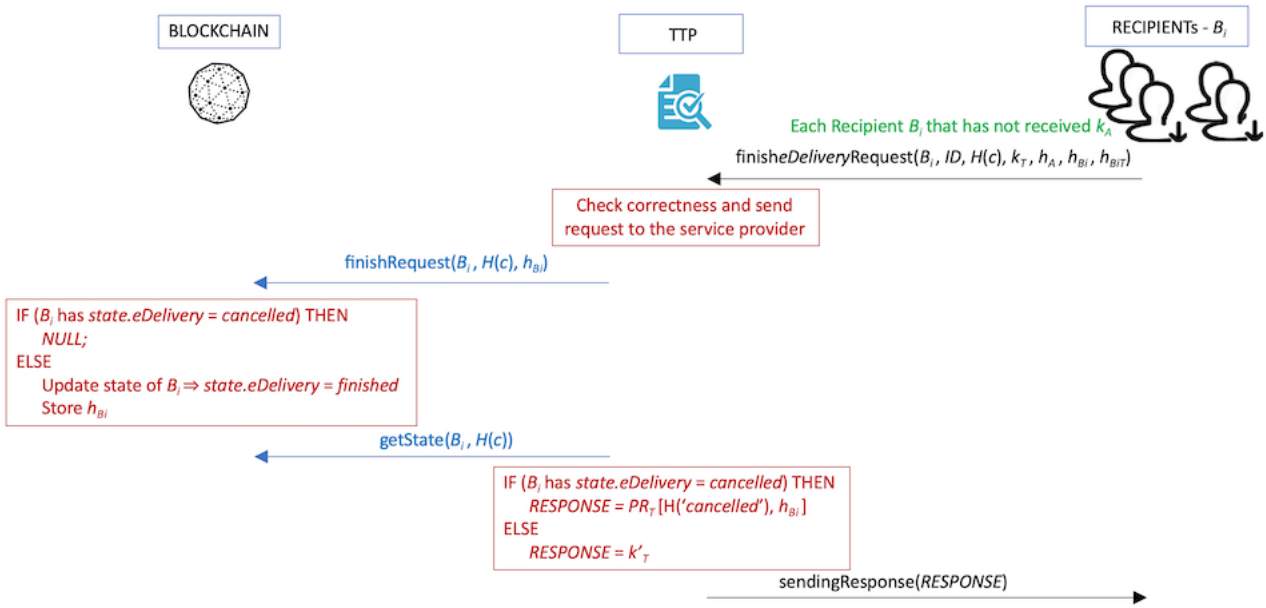

FIGURE 6. On-chain finish subprotocol.

```
enum State {notexists, created, cancelled, accepted,
     finished, rejected}

address public sender;
address[] public receivers;
mapping (address => State) public receiversState;
uint acceptedReceivers;

bytes32 public messageHash;
string public message;

uint public term1;
uint public term2;
uint public start;
```

Listing 1. Variables.

Data structures are required to store the set of receivers and provide multiparty capabilities, so an array stores the list of addresses of the receivers, and a mapping stores the state of the exchange for each receiver. As it has been explained in Section §VI, in this multiparty protocol the state of the exchange depends on each receiver. We use a mapping to get a constant time search, and, at the same time, a constant cost search. But mappings are not iterable. For this reason, we have also an array to store all addresses of the receivers, to iterate through all these addresses.

The contract also needs to save the *messageHash* and the *message* values of the eDelivery and use two periods: *term1* and *term2*. The *start* variable stores the time when the delivery is created, and are used to set the timeouts. *acceptedReceivers* will count the number of receivers that have accepted, to let the sender finish the delivery, avoiding the waiting until the deadline of *term1*. The use of this variable will avoid the need to check the state of all receivers, that is expensive in terms of gas. This data structures can be seen in Listing 1.

```
function accept() public {
  require(now < start+term1, "The timeout term1 has been
     reached");
  require(receiversState[msg.sender]==State.created, "
     Only receivers with 'created' state can accept");

  acceptedReceivers = acceptedReceivers+1;
  receiversState[msg.sender] = State.accepted;
}
```

Listing 2. Accept function.

When a new *Smart Contract* of this type is created, an array of receivers are set with the parameters passed by the sender, and then the state of every receiver of the delivery are initialized to *created*.

The *Accept* function (see Listing 2) checks that the sender of the function is in the receivers mapping, and its delivery state is *created*. It also checks that the deadline of *term1* has not been reached. If all these conditions are satisfied, the delivery state of this receiver is set to *accepted*.

The *Finish* function (see Listing 3) checks that the deadline of *term1* has been reached or if all receivers have been accepted. The function also verifies that its caller is the sender of the eDelivery, and that the text of the message matches the hash specified before. If all of this conditions are fulfilled, the delivery states of the receivers that were *accepted* are set to *finished*, and the states of the receivers that were *created* are set to *rejected*.

Finally, the *Cancel* function (see Listing 4) checks that the *term2* timeout has been reached and that the function is called by a receiver with delivery state *accepted*. If all these conditions are satisfied, the receiver's delivery state is set to *cancelled*.

In order to create and deploy reusable registered eDelivery contracts we use a common used design pattern, a *Factory*

```
function finish(string _message) public {
  require((now >= start+term1) || (acceptedReceivers>=
     receivers.length), "The timeout term1 has not been
     reached and not all receivers have been accepted the
     delivery");
  require (msg.sender==sender, "Only sender of the
     notification can finish");
  require (messageHash==keccak256(_message), "Message not
     valid (different hash)");

  message = _message;
  sender.transfer(this.balance);
  for (uint i = 0; i<receivers.length; i++) {
    if (receiversState[receivers[i]] == State.accepted) {
      receiversState[receivers[i]] = State.finished;
    } else if (receiversState[receivers[i]] == State.
     created) {
      receiversState[receivers[i]] = State.rejected;
    }
  }
}
```

Listing 3. Finish function.

```
function cancel() public {
  require(now >= start+term2, "The timeout term2 has not
     been reached");
  require(receiversState[msg.sender]==State.accepted, "
     Only receivers with 'accepted' state can cancel");

  receiversState[msg.sender] = State.cancelled;
}
```

Listing 4. Cancel function.

Contract, a type of contract that creates and deploys other contracts. With this pattern we achieve the reusability of the Non-Confidential Multiparty Registered eDelivery contracts, and we get the following advantages:

- The factory can be used to store the addresses of the child contracts, so that they can be extracted whenever necessary. This is safer than storing it in an external database, preventing the loss of references to these contracts.
- Our front-end service only needs to store the address of the factory contract.
- The front-end service has to pay only for the deployment of this factory contract.
- Users could invoke a function in the factory to deploy a new delivery contract, paying deployment costs (see Section §X). Then, the factory itself deploys the delivery contract and stores the address of the deployed contract in an internal list.

B. SMART CONTRACT FOR THE CONFIDENTIAL MULTIPARTY REGISTERED eDELIVERY PROTOCOL

In the Confidential Multiparty blockchain-based protocol, the sender A and the set of receivers B will exchange messages and non-repudiation evidence directly. Only as a last resort, in the case they cannot get the expected items from the other party, the smart contract or the TTP would be invoked, by calling their *cancel* or *finish* functions.

A *Smart Contract* based on the smart contract presented in [23] is used, adapted to the new protocol, so now there is a possibility for a message to be sent to several receivers. A data structure allows the association of multiple receivers to the same message and is in charge of storing the necessary parameters and determining the state of the exchange.

```
enum State {notexists, cancelled, finished }

struct ReceiverState{
  bytes32 receiverSignature;
  bytes32 keySignature;
  State state;
}

struct Message{
  bool messageExists;
  address sender;
  address[] receivers;
  mapping(address => ReceiverState) receiversState;
}

mapping(address => mapping(uint => Message)) messages;
address public ttp;
```

Listing 5. Variables and constructor.

```
function cancel(uint _id, address[] _receivers) public {
  for (uint i = 0; i<_receivers.length;i++){
    address receiverToCancel = _receivers[i];
    if (messages[msg.sender][_id].receiversState[
     receiverToCancel].state==State.notexists) {
      addReceiver(_id, msg.sender, receiverToCancel, 0,
     0, State.cancelled);
    }
  }
}
```

Listing 6. Cancel function.

```
function finish(uint _id, address _sender, address
     _receiver, bytes32 _receiverSignature, bytes32
     _keySignature) public {
  require (msg.sender==ttp, "Only TTP can finish");
  if (messages[_sender][_id].receiversState[_receiver].
     state==State.notexists) {
    addReceiver(_id, _sender, _receiver,
     _receiverSignature, _keySignature, State.finished);
  }
}
```

Listing 7. Finish function.

The state of the exchange is no longer uniquely defined in the message, but depends on each receiver. The set of receivers will be stored within the *struct* message by using a *mapping*, as can be seen in the code in Listing 5. In the *ttp* variable, we will store the address of the account that creates this smart contract.

Function *Cancel* (Listing 6) uses as the input parameter an array that contains the set B''. The function goes through this set and cancels the exchange for each receiver, if it does not exist previously.

Function *Finish* (Listing 7) checks if B_i exists. If B_i does not exist, the TTP includes it in *B"-finished*, and the NRR proof h_{Bi} and the key k, with its corresponding private key, are stored.

The smart contract also includes a function for the addition of a new receiver (Listing 8), that also checks if that message exists and creates it if necessary.

It is important to stress that unlike the non-confidential protocol, in the confidential protocol we don't use the *Factory Contract* pattern to create new deliveries. This is due to the fact that in this protocol we only need to store simple variables like addresses, states and binary signatures, in contrast to the non-confidential protocol, where we need to use more complex structures and functions for each delivery. For this

```
function addReceiver(uint _id, address _sender, address
    _receiver, bytes32 _receiverSignature, bytes32
    _keySignature, State _state) private {
  if (!messages[msg.sender][_id].messageExists) {
    messages[_sender][_id].sender = _sender;
    messages[_sender][_id].messageExists = true;
  }
  messages[_sender][_id].receivers.push(_receiver);
  messages[_sender][_id].receiversState[_receiver].
    receiverSignature = _receiverSignature;
  messages[_sender][_id].receiversState[_receiver].
    keySignature = _keySignature;
  messages[_sender][_id].receiversState[_receiver].state
    = _state;
}
```

Listing 8. addReceiver function.

reason, we can store all this in a single smart contract, using mappings and structs, that is cheaper than deploy an smart contract for every single delivery.

IX. EVALUATION OF PROPERTIES

This section presents an analysis of the ideal properties of a registered *eDelivery* system (listed in Section II) for both proposals. The basis of this rationale about the security and privacy properties is the correct use of the cryptographic primitives. Our proposals make use of the following primitives:

- Digital Signatures
- Symmetric Encryption
- Public Key Encryption
- Key Wrapping
- Hash Functions

Thus, in order to make a secure implementation, any deployment has to take into consideration the official documents that address the use of cryptographic algorithms and which key lengths are specified. The last issue of the NIST document [33] includes the explanation of the projected maximum-security strength of key lengths associated to the cryptographic algorithms. Also, the document has prediction of the period of time throughout the algorithms and the proposed key lengths are expected to provide probable security. That is to say, in order to break the security any attacker must solve the underlying problem of the implemented cryptographic operations. Thus, the protocols provide suitable and practice-oriented provable security as far as implementors chose sufficiently large values of the security parameters and key lengths according to the international standards [33]. Regarding the specific cryptographic operations used in our protocols, the digital signature algorithms [34], key wrapping and public key encryption operations are based on the length and the proper generation of the domain parameters used.

In our proposals we have both operations that are performed off-chain and on-chain. For the off-chain operations, in order to provide *acceptable* security (i.e. no security risk is currently known when used in accordance with any associated guidance), the DSA domain parameter lengths has to be (2048, 224) or (2048, 256), which provide a security strength of 112 bits; or (3072, 256), which provides a security strength of 128 bits [33]. Finally, hash function family SHA-2 specified in [35] provides *acceptable* security for all hash function applications. Regarding on-chain communication, signatures are used to authorize transactions on behalf of the signer. They are also used to prove to a smart contract that a certain account approved a certain message. Therefore, these signed messages can be used as non-repudiation evidence. Well-known blockchains such as Bitcoin and Ethereum apply the ECDSA algorithm to create signatures on transactions. In particular, Ethereum signatures use ECDSA and secp256k1 constants to define the elliptic curve and create secure signatures [36]. Providing that developers follow the standard guidelines to use this cryptography in the implementation of our protocol, the following discussion holds.

The discussion about security includes six proposition to evaluate properties: *effectiveness*, *fairness and evidence transferability*, *temporal parameters* (timeliness and timestamping), *non-repudiation*, *trusted third parties* (presence, verifiability and maintenance of state information) and *confidentiality*. Each proposition is formed by different claims with its respective proofs and a final result.

The property of *efficiency* is not included in this analysis because we have performed a list of experiments to evaluate the efficiency of the protocols, and their results are included in Section X.

Proposition 1 (Effectiveness): The proposed protocols for registered eDelivery are effective, that is, if the parties behave correctly, they will receive the expected items.

Claim 1: The *non-confidential* protocol for multiparty registered eDelivery is *effective*.

Proof: In order to send a new *eDelivery*, the sender creates a new instance of the smart contract to perform the specified functions according to the regular operational mode of the specified protocol. If all parties invoke all the functions correctly, all of them will receive the expected items, as it is easily deduced from the smart contract solidity code included in Listing 1, Listing 2, Listing 3 and Listing 4 and in more detail in our github repository. Actually, no *TTP* will be involved in the protocol in any case, even if any party does not follow the three-step exchange protocol, the smart contract will ensure a fair result for each party:

- If the receiver does not invoke the *accept()* function, then the sender can invoke the *finish()* function to solve the exchange.
- If the sender does not invoke the *finish()* function, then the receiver can invoke the *cancel()* function to solve the exchange.

Claim 2: The *confidential* protocol for multiparty registered eDelivery is *effective*.

Proof: This protocol has an optimistic approach for an *eDelivery* issuing, thus if the parties correctly execute the off-chain steps specified in Section VII-A the exchange will successfully conclude without any intervention of the *TTP* and will produce the desired result for each one. Only

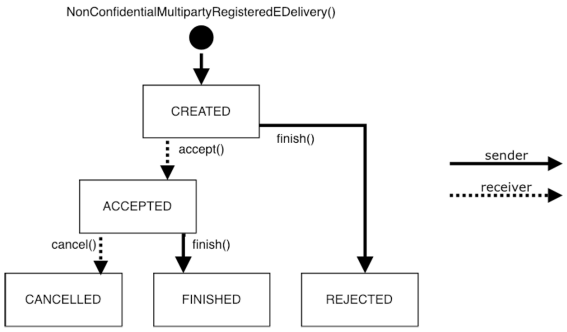

FIGURE 7. Life cycle of the states in the non-confidential notifications protocol.

if the sender does not execute the step number 3 of this subprotocol, the receiver has to invoke the *TTP* to conclude the exchange. The first step uses public key encryption while all the steps include digital signatures. All these operations take into account the secure use of cryptographic primitives included in the beginning of this section.

Result 1: According to what is said in Claim 1 and Claim 2, the proposed *eDelivery* protocols fulfill the property of effectiveness.

Proposition 2 (Fairness and Evidence): The proposed protocols for registered *eDelivery* are fair, so after completion of a protocol run, either each party receives the expected item or neither party receives any useful information about the other's item. Moreover, in the proposed protocol for non-confidential registered *eDelivery*, the proofs generated by the system can be transferred to external entities to prove the result of the exchange.

Claim 3: The multiparty *non-confidential* registered eDelivery protocol provides *strong fairness*.

Proof: On the one hand, according to the protocol described in Section VI, the sender A will not receive the non-repudiation proof of reception provided by the smart contract unless she makes the transaction that registers the message on the blockchain (case *state.eDelivery=finished*). On the other hand, a recipient B_i will not have access to the message unless he executes a transaction to accept the *eDelivery* (*state.eDelivery=accepted*). At any moment, the smart contract does not generate alternative cancellation or finalization proofs that could create any situation where one of the parties could have contradictory proofs (thus, there is no action that can lead the exchange to *weak fairness* situation), as can be seen in Figure 7.

Claim 4: The generated proofs in the multiparty *non-confidential eDelivery* protocol can be presented as *evidence* to an external entity.

Proof: Since the parties cannot obtain contradictory proofs in any way (evidence can be only generated by the logic of the smart contract), the generated proofs can be presented as evidence to an external entity. Moreover, its transferability its easy, since the results of the exchange are stored on the blockchain. Due to the immutability of the blockchain, the content of the message cannot be modified so the system provides integrity to the message. The moment the delivery takes place can be derived from the timestamp of the block where the transaction was included.

Claim 5: The multiparty *confidential* registered *eDelivery* protocol provides *weak fairness*.

Proof: The confidential protocol, described in Section VII, does not allow any party to receive the expected item from the other party unless the latter is in position to get access to the correspondent expected item. However, the intervention of the *TTP* can lead to a situation where one of the parties possesses contradictory evidence. For instance, a malicious sender A can have the non-repudiation proof received directly from the recipient B and, in addition, she can get the cancellation proof generated by the smart contract after invoking the correspondent cancellation request to the smart contract. For this reason, the fairness will be weak and the generated proofs are non transferable.

Claim 6: The *confidential eDelivery* protocol proposed in Section VII has not the property of transferability although the *evidence* of the final state of the exchange can be consulted in the blockchain.

Proof: The sender can get a non-repudiation proof from the recipient in the step 2 of the exchange protocol (Section VII-A). However, this is not enough to certainly prove to a third party that the recipient has received the message. Additionally, the third party should check the information stored on the blockchain related to this particular *eDelivery* to confirm this issue. Thus, this protocol does not provide transferable evidence, since a protocol generates transferable evidence only if sender and recipient can separately demonstrate to any third party the result of the exchange without the need to request other entities.

Result 2: In the non-confidential protocol, the evidence can only be generated by the smart contract. Thus, according to what is stated in Claim 3 and Claim 4, the smart contract will guarantee the fairness of the exchange (*strong fairness*) and provides transferability of evidence. In contrast, in the confidential protocol, actors can get evidence from the exchange protocol and from the smart contract. Thus, if a claim arises, the arbitrator has to consult both parties to resolve the final state of the exchange (*weak fairness*).

Proposition 3 (Temporal Parameters): The multiparty registered *eDelivery* approaches in Sections VI and VII offer timestamp and timeliness.

Claim 7: The multiparty *non-confidential* registered *eDelivery* protocol satisfies *weak timeliness*.

Proof: The protocol is not asynchronous. If one of the parties delays its intervention in the exchange, the other party will not be able to resolve it until the deadline. However, after the deadline both parties can request the finalization of the exchange (see Figure 3, Listing 3 and Listing 4). Moreover, the protocol wants to motivate the sender to conclude the exchange before the timeout blocking an amount of money in the smart contract. This amount will only be refunded to the sender if she concludes before the deadline. All the transactions performed on the blockchain are timestamped.

Claim 8: The multiparty *confidential* registered *eDelivery* protocol fulfills the *strong timeliness* property.

Proof: The parties can finish the exchange at any moment accessing the smart contract (sender *A*) or contacting the *TTP* (receiver *B*). The duration of the resolution will depend of the block notification treatment. The protocol can assume that transactions are valid immediately (zero confirmation) or wait until the block is confirmed in the chain (fully confirmation). All the transactions performed on the blockchain are timestamped.

Result 3: The multiparty confidential registered *eDelivery* protocol fulfills the *strong timeliness* property, since either party can invoke the correspondent finalization procedure at any moment. But the non-confidential protocol satisfies *weak timeliness*, since parties cannot decide to finish the exchange sooner than the specific timeouts.

Proposition 4 (Non-Repudiation): In the proposed protocols for registered *eDelivery* any sender cannot deny being the origin of an item and any recipient cannot deny being the receiver of an item either.

Claim 9: The multiparty *non-confidential* registered *eDelivery* protocol achieves *non-repudiation of origin* together with *non-repudiation of receipt* after the execution of the exchange.

Proof: Regarding the sender *A*, she cannot deny having sent the message since there is a transaction on the blockchain from her address containing the message and another one related with the same message including the address of the receiver and the hash of the message. With respect to the recipient *B*, he cannot deny having received the delivered data since there is a transaction from his address in the blockchain accepting the reception of the message and the *State* of the exchange is *Finished*, so the message is publicly accessible in the blockchain.

Claim 10: The multiparty *confidential* registered *eDelivery* protocol provides *non-repudiation of origin* together with *non-repudiation of reception* evidence to the involved parties in an exchange.

Proof: The protocol achieves non-repudiation of origin together with non-repudiation of receipt after a successful execution of the three-step exchange where secure cryptographic primitives are used or, if this subprotocol does not finish successfully, then users can complete the exchange by means of the smart contract. Thus, sender *A* cannot deny having sent the message since recipient *B* has the element received in the third step of the protocol (signed by *A*) or the state of the smart contract is *Finished*. In addition to that, *B* cannot deny having received the message since *A* has the elements sent by *B* in the second step of the protocol.

Result 4: As it is stated in 9 and 10 , senders and recipients will get the correspondent non-repudiation evidence from the proposed *eDelivery* protocols. Thus, *eDelivery* schemes proposed in this paper achieve the non-repudiation property.

Proposition 5 (Trusted Third Parties): The protocols proposed in this paper are verifiable. A *TTP* is only used in the multiparty confidential protocol. The involvement of the third party in this protocol can be verified and it is not required that the *TTP* maintains state information.

Claim 11: There is a total absence of *TTP* in the multiparty *non-confidential eDelivery* protocol. It has been substituted completely by the smart contract, but even so the actions performed in the protocol are verifiable.

Proof: This proposal does not require an external party acting as a *TTP*. Parties execute the functions of the smart contract creating the associate transactions and there is no need of dispute resolution. All the communications in this protocol are on-chain, thus, they are stored on the blockchain. Blockchain has been designed to be immutable and publicly verifiable, therefore the actions completed in the protocol can be verified and anyone can know which address is accountable for that.

Claim 12: In the multiparty *confidential eDelivery* protocol, the *TTP* is only involved on case of exception. This third party is an optimistic stateless *TTP*. Moreover, the blockchain and the smart contract provide evidence of the *TTP* intervention.

Proof: The *TTP* is not involved in the exchange subprotocol described in Section VII. The *TTP* only intervenes in the protocol if the exception case: when the recipient claims that she has not received the expected item from the sender, according to the first step of the exchange protocol. When the *TTP* is involved in the dispute resolution, it can resolve the exchange through the use of the smart contract, executing the *finish* function. The *TTP* does not need to store any kind of state information of the exchange.

The *TTP* is only able to invoke the *finish()* function of the smart contract according to the data provided by the recipient of the *eDelivery* (the recipient will provide the *sender address* and the identifier to identify an *eDelivery*). Then, the *TTP* will answer to the recipient's request as the smart

contract stated. If the *TTP* misbehaves, anyone who knows the *eDelivery* identifiers (i.e. *sender address* and message identifier) can verify the answer of the *TTP* according to the data stored on the blockchain.

Result 5: Thanks to the blockchain technology there is no need of *TTP* involvement in the non-confidential *eDelivery* protocol. Nevertheless, the protocol is verifiable. In the multiparty confidential protocol the *TTP* will only be involved if the recipient claims that she has not received the appropriate item from the sender. In this case, the *TTP* will act according to what is established in the smart contract, otherwise the blockchain can provide evidence of any wrong behavior. Thus, the actions of the *TTP* are verifiable. Also, the blockchain allows to keep public and verifiable the state of any *eDelivery* that uses the confidential scheme, therefore the *TTP* involved in the exchange is *stateless*.

Proposition 6 (Confidentiality): Confidentiality is an optional property in registered *eDelivery* protocols. The protocols proposed in this paper provide solutions for both confidential and non-confidential (or public) deliveries.

Claim 13: Since *confidentiality* is an optional property, the multiparty *non-confidential* registered *eDelivery* protocol does not implement this feature, allowing public deliveries.

Proof: Instead of keeping the message secret, the protocol specified in Section VI stores the data related to any exchange on the blockchain. Thus, it offers the possibility to access the message of any registered *eDelivery* though a blockchain explorer.

Claim 14: The multiparty *confidential* registered *eDelivery* protocol assures the *confidentiality* of the delivered data.

Proof: If the exchange is finished through the execution of the three-step exchange subprotocol, then no other entity is involved in the exchange, and the message remains confidential (the message is encrypted using a symmetric cipher prior to sending it). If the *TTP* is involved or the functions of the smart contract are executed, then the *TTP* will process the received elements and will make a transaction including the element that will allow the recipient *B* to decrypt the message but the plain message is not included in the transaction so it will not be included in a block of the blockchain to preserve the confidentiality.

Result 6: The confidential blockchain-based protocol of Section VII can be used if the confidentially of the message is desired in an *eDelivery* exchange. Otherwise, when the message has to be public, the scheme of the Section VI is suitable.

X. PERFORMANCE ANALYSIS

This performance analysis includes experiments to determine the efficiency of the system in terms of cost, since the economical execution costs could be a concert in the development of the *eDelivery* service. These tests have been performed using the *Smart Contracts* explained in Section §VIII, deployed in the Ganache network, a personal blockchain used for Ethereum development, to isolate the performance conditions and possible problems of a real network like the main Ethereum network or the Rinkeby test network. For both Non-Confidential and Confidential protocols, we have detailed the gas cost of their main functions, comparing the Two-Party version with the Multiparty protocol for one, two and ten receivers. With this tests we have two main objectives: in the one hand we want to obtain absolute values of the economical costs to evaluate the viability of the protocols and in the other hand we want to evaluate if the multiparty protocols are able to reduce the cost of the Two-party protocols.

We have also tested the performance in terms of delay, to have an illustrative reference of the delays introduced by the functions, although we have not included the complete list of results in this paper. In the Ganache network all the functions have been executed with a delay between 35 and 170 milliseconds, closely related to the gas cost of each function. But in a normal production Ethereum blockchain network, all the transactions need a few seconds to be validated through the mining process, which have a delay of 15-30 seconds, this is, the greatest component of the delay value.

Regarding the protocols, we can say that they use the minimum number of steps that allow the effective exchange. In the confidential protocol, moreover, the parties can finalize the exchange without the need to contact with a TTP or execute any function of the smart contract. If the parties do not follow the protocol and the execution of the smart contract is required, the gas necessary for its operation would be reduced compared with the protocol for non-confidential registered eDelivery protocol.

The exact value of the execution cost will be useful to check when a multiparty version of a protocol would be more efficient than a two-party protocol. Moreover, the results will be also useful to compare the cost of the non-confidential and the confidential protocols.

In Figure 8 we can see the gas cost of the main functions of the Non-Confidential two-party and multiparty protocol. From their analysis we can conclude:

- The deploy of the factory and the deploy of the delivery (*createDelivery()* function) are considerably more expensive than the *accept()* and *finish()* functions.
- We have to take into account that the cost of the two more expensive operations (deploy and *createDelivery* function) is distributed between the owner of the service (the *Factory* deployer) and the sender of the delivery.
- It is cheaper to deploy a two-party delivery than a Muliparty delivery with only one receiver. But when the same data have to be sent to two or more receivers, it is cheaper to do it with the Multiparty protocol, because it avoids the deployment of one smart contract for each receiver.

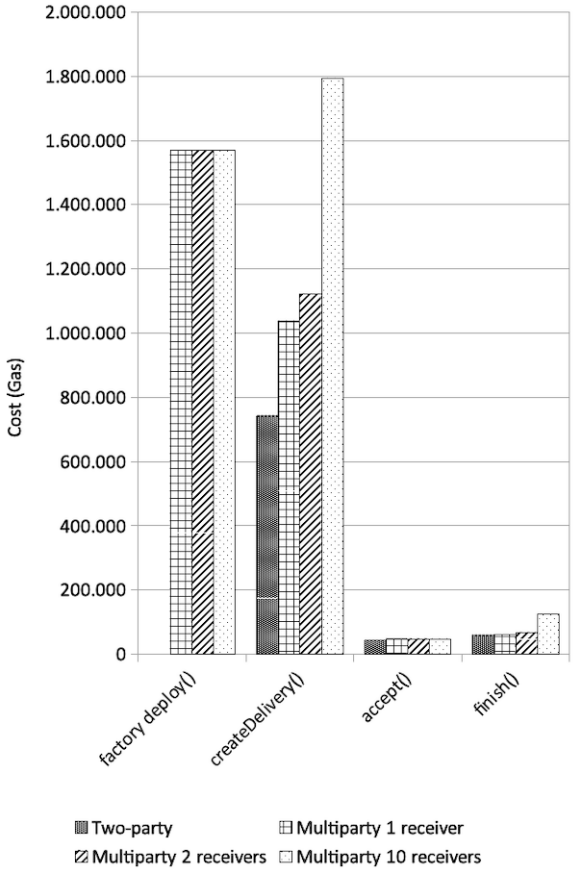

FIGURE 8. Non-confidential multiparty protocol cost.

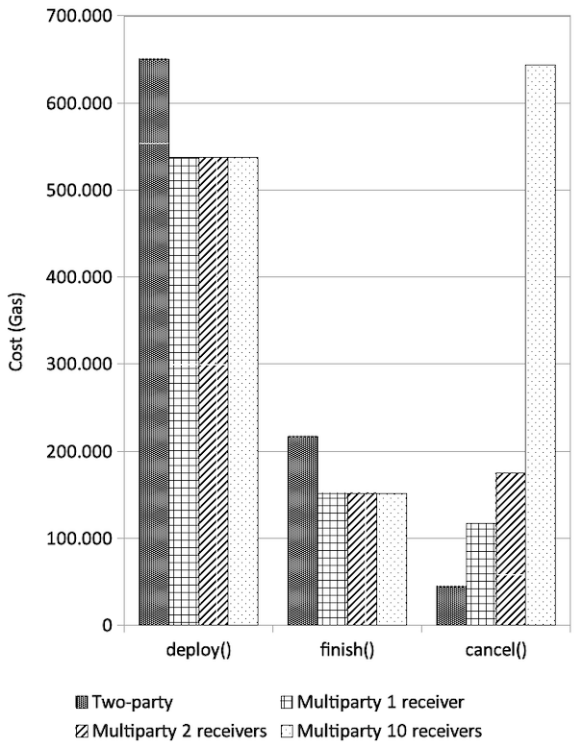

FIGURE 9. Confidential multiparty protocol cost.

In Figure 9 we can see the performance in terms of gas cost of the Confidential two-party and multiparty protocols. We can highlight the following conclusions:

- The gas cost is, in general cheaper than the Non Confidential protocol, but we have to consider that the smart contract of the Confidential protocol will be used only when the TTP is required. The rest of this protocol functionalities are out of the blockchain.
- The *deploy()* and *finish()* functions are cheaper in the Multiparty protocol than in the Two-party protocol because we have optimized the code presented in [23] for the two-party protocol, making the multiparty protocol more efficient in any case.
- With respect to the Multiparty protocol, the *deploy()* and *finish()* functions cost the same because the former do not depend on the number of receivers, and the latter can only finish the protocol for one receiver. On the other hand, in the *cancel()* function we can cancel the delivery for a variable number of receivers.

The cost of this analysis is computed in gas, but to know the exact price of this transactions we also need the gas price, set in Ethers, and the Ethers to US-Dollars exchange rate, but neither one nor the other are fixed. For this reason, we have added to Table 3 and Table 4 the price in US-Dollars of each of the functionalities as a guideline, considering the exchange rate at February 21, 2019, and taking into account a gas price of 1 Gwei and 20 Gwei. The main difference, in addition to the total cost of the transaction, is the time it will take the transaction to be accepted by a mining node. Thus, a transaction made on the main Ethereum network with a value of 1 Gwei can take almost 100 minutes (depending on the current network traffic), while in the case of 20 Gwei can be reduced to 30 seconds.[4]

TABLE 3. Non confidential multiparty protocol cost in gas and equivalent US-Dollars price with 1 Gwei - 20 Gwei gas price.

Cost in Gas	Two-party	Multiparty 1 receiver	Multiparty 2 receivers	Multiparty 10 receivers
factory deploy()	0	1,568,284 (0.23-4.58)	1,568,284 (0.23-4.58)	1,568,284 (0.23-4.58)
create-Delivery()	742,569 (0.11-2.17)	1,038,402 (0.15-3.03)	1,122,305 (0.16-3.28)	1,793,522 (0.26-5.23)
accept()	43,545 (0.01-0.13)	47,711 (0.01-0.14)	47,711 (0.01-0.14)	47,711 (0.01-0.14)
finish()	59,489 (0.01-0.17)	60,642 (0.01-0.18)	67,725 (0.01-0.20)	124,389 (0.02-0.36)

TABLE 4. Confidential multiparty protocol cost in gas and equivalent US-Dollars price with 1 Gwei - 20 Gwei gas price.

Cost in Gas	Two-party	Multiparty 1 receiver	Multiparty 2 receivers	Multiparty 10 receivers
deploy()	650,451 (0.09-1.90)	537,397 (0.08-1.57)	537,397 (0.08$- 1.57$)	537,397 (0.08$- 1.57$)
finish()	216,921 (0.03-0.63)	151,697 (0.02-0.44)	151,674 (0.02$- 0.44$)	151,687 (0.02$- 0.44$)
cancel()	44,462 (0.01-0.13)	116,385 (0.02-0.34)	174,950 (0.03$- 0.51$)	643,982 (0.09$- 1.88$)

[4]https://ethgasstation.info/

TABLE 5. Comparison of properties.

Property	Non-Confidential Multiparty Protocol	Confidential Multiparty Protocol
Effectiveness	YES	YES
Fairness	STRONG	WEAK
Timeliness	WEAK	STRONG
Non-repudiation	YES	YES
Verificability	YES	YES
Confidentiality	NO	YES
Evidence Transferibility	YES	NO
TTP	VOID	OPTIMISTIC STATELESS

XI. COMPARISON AND CONCLUSIONS

Existing legislation lays down the rules for electronic identification and trust services for electronic transactions. Then technical proposals must achieve the legal requirements to be qualified. The features of Qualified Electronic Registered Delivery, one of the trust services included in the regulation, are similar to those offered by fair exchange protocols: non-repudiation of origin and reception together with integrity of the data. For this reason it is possible to design a fair exchange protocol for registered eDelivery. However, this kind of services usually rely heavily on the use of trusted third parties and are costly and inefficient and the behavior of the TTP has to be verified.

We have proved that the blockchain-based technologies can be very useful in the design of a qualified registered eDelivery service, solving the problems related with the use of TTPS.

The main concluisons of the new solution are:

- It is possible to create both a solution that registers the delivery and also a proposal that protects the confidentiality of the delivered data. The choice affects strongly the design of the protocol.
- A multiparty protocol is much more efficient than a two-party protocol. The performance analysis show that even for only two receivers the multiparty protocol requires less gas than the use of the two-party protocol.
- Using smart contracts it is possible to achieve the ideal properties without the intervention of a TTP or with a minimal involvement.
- How, when and by whom a smart contract is deployed depends on the protocol. Another difference is who pays for the service.
- The use of a factory to deploy the smart contract is well suited for the non-confidential protocol but is useless is the confidential protocol. Instead, this protocol makes use of message identifiers.
- The results of the analysis of properties prove that the protocols achieve the ideal properties of the service: effectiveness, fairness, timeliness, non-repudiation, verificability and confidentiality, as it is summarized in Table 5.
- The results of the performance analysis are useful to compare the protocols in terms of efficiency. The two proposed protocols differ in the amount of gas required to execute the functions of the smart contracts. Moreover, we have to take into account that some functions are not mandatory to finish the exchange.
- The price to execute the smart contract and the time required for the validation of the transactions are related to the gas price. This way, for a low gas price, the eDelivery can be executed with a cost of only a few cents of dollar. However, the delay will be greater than an hour. Incrementing the gas price we have obtained validation times of a few seconds, while the execution costs are increased to more than a dollar. As a conclusion, the delay and the cost of the eDelivery can be controlled adjusting the gas price.
- Blockchain-based technologies have been proved useful in the design of fair exchange protocols, so it may be used for the design of protocols for different services.
- The role of trusted third parties can be affected by the incorporation of blockchain-based technologies in the design of protocols.

As future work, we propose four improvements:

- The proposals allow the delivery between parties identified by its blockchain addresses. In the future we will integrate a blockchain-based identity system to make deliveries between user identities.
- We have performed delay tests on a local network, using Ganache. They have provided coherent results to compare the protocols, but as future work we intend to calculate the delays on real blockchain networks.
- In the literature, some of the definitions of the ideal properties for fair exchange protocols are made taking into account a traditional scenario, using TTPs. Since different services can benefit from the approach presented in this paper and more blockchain-based protocols for fair exchange can be designed in the future, we will work in the redefinition of the properties for fair exchange protocols when blockchain technologies are used.
- Since the protocols are independent of the platform used for their implementation, we will work in the improvement of the efficiency and reduction of economic costs trying implementations on different platforms.

REFERENCES

[1] N. Asokan, V. Shoup, and M. Waidner, "Asynchronous protocols for optimistic fair exchange," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 1998, pp. 86–99.

[2] S. Barber, X. Boyen, E. Shi, and E. Uzun, "Bitter to better—How to make bitcoin a better currency," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 399–414.

[3] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in *Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun., IEEE 14th Int. Conf. Smart City, IEEE 2nd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2016, pp. 1392–1393.

[4] S. A. Chaudhry, M. S. Farash, H. Naqvi, and M. Sher, "A secure and efficient authenticated encryption for electronic payment systems using elliptic curve cryptography," *Electron. Commerce Res.*, vol. 16, no. 1, pp. 113–139, 2016.

[5] S. Delgado-Segura, C. Perez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomarti, "A fair protocol for data trading based on Bitcoin transactions," *Future Gener. Comput. Syst.*, to be published.
[6] *Electronic Signatures and Infrastructures (ESI); Registered Electronic Mail (REM) Services*, document ETSI EN 319 532-5, ETSI, 2018.
[7] *Security Guidelines on the Appropriate Use of Qualified Electronic Registered Delivery Services*, Eur. Union Agency Netw. Inf. Secur., Heraklion, Greece, Dec. 2016.
[8] H. Ethan, F. Baldimtsi, L. Alshenibr, A. Scafuro, and S. Goldberg, "TumbleBit: An untrusted tumbler for bitcoin-compatible anonymous payments," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2017, pp. 1–15.
[9] J. L. Ferrer-Gomila, M. Payeras-Capellà, and L. I. Huguet i Rotger, "An efficient protocol for certified electronic mail," in *Proc. Inf. Secur. Workshop (ISW)*, 2000, pp. 237–248.
[10] J. L. Ferrer-Gomila, M. Payeras-Capellà, and L. I. Huguet i Rotger, "A realistic protocol for multi-party certified electronic mail," in *Proc. Inf. Secur. Conf. (ISC)*, 2002, pp. 210–219.
[11] J. L. Ferrer-Gomila, J. A. Onieva, M. Payeras, and J. Lopez, "Certified electronic mail: Properties revisited," *Comput. Secur.*, vol. 29, no. 2, pp. 167–179, 2010.
[12] S. Goldfeder, J. Bonneau, R. Gennaro, and A. Narayanan, "Escrow protocols for cryptocurrencies: How to buy physical goods using bitcoin," in *Proc. 21st Int. Conf., Financial Cryptogr. Data Secur. (FC)*, in Lecture Notes in Computer Science, vol. 10322, 2017, pp. 321–339.
[13] H. R. Hasan and K. Salah, "Blockchain-based solution for proof of delivery of physical assets," in *Blockchain* (Lecture Notes in Computer Science), vol. 10974. Seattle, WA, USA: Springer, Jun. 2018, pp. 139–152.
[14] H. Hasan and K. Salah, "Proof of delivery of digital assets using blockchain and smart contracts," *IEEE Access*, vol. 6, pp. 65439–65448, 2018.
[15] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2016, pp. 43–60.
[16] Q. Huang, G. Yang, D. S. Wong, and W. Susilo, "A new efficient optimistic fair exchange protocol without random oracles," *Int. J. Inf. Secur.*, vol. 11, no. 1, pp. 53–63, 2012.
[17] Q. Huang, D. S. Wong, and W. Susilo, "P^2OFE: Privacy-preserving optimistic fair exchange of digital signatures," in *Topics in Cryptology*. San Francisco, CA, USA: Springer, 2014, pp. 367–384.
[18] H. Kılınç and A. Küpçü, "Optimally efficient multi-party fair exchange and fair secure multi-party computation," in *Topics in Cryptology* (Lecture Notes in Computer Science), vol. 9048. San Francisco, CA, USA: Springer, Mar. 2015, pp. 330–349.
[19] S. Kremer and O. Markowitch, "A multi-party non-repudiation protocol," in *Proc. IFIP Conf.* Kluwer: Amsterdam, The Netherlands, 2000, pp. 271–280.
[20] S. Kremer and O. Markowitch, "Selective receipt in certified e-mail," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2247. New York, NY, USA: Springer-Verlag, 2001, pp. 136–148.
[21] J. Liu, W. Li, G. Karame, and N. Asokan, "Toward fairness of cryptocurrency payments," *IEEE Secur. Privacy*, vol. 16, no. 3, pp. 81–89, Jun. 2018.
[22] M. Mut-Puigserver, J. Ferrer-Gomila, and L. Huguet-Rotger, "Certified e-mail protocol with verifiable third party," in *Proc. IEEE Int. Conf. e-Technol., e-Commerce e-Service*, 2005, pp. 548–551.
[23] M. Mut-Puigserver, M. Payeras-Capellà, and M. Cabot-Nadal, "Blockchain-based fair certified notifications," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (Lecture Notes in Computer Science), vol. 11025. Barcelona, Spain: Springer, 2018, pp. 20–37.
[24] J. Onieva, J. Zhou, and J. Lopez, "Enhancing certified email service for timeliness and multicasting," in *Proc. 4th Int. Netw. Conf.*, 2004, pp. 327–336.
[25] J. A. Onieva, J. Zhou, and J. Lopez, "Multiparty nonrepudiation: A survey," *ACM Comput. Surv.*, vol. 41, no. 1, pp. 5:1–5:43, Jan. 2009.
[26] M. Payeras-Capellà, M. Mut-Puigserver, J. Ferrer-Gomila, and L. Huguet-Rotger, "No author based selective receipt in an efficient certified e-mail protocol," in *Proc. Parallel Distrib. Process. (PDP)*, Feb. 2009, pp. 387–392.
[27] M. Payeras-Capellà, M. Mut-Puigserver, and M. Cabot-Nadal, "Smart contract for multiparty fair certified notifications," in *Proc. 6th Int. Symp. Comput. Netw.*, Takayama, Japan, 2018, pp. 459–465.
[28] Z. Shao, "Fair exchange protocol of Schnorr signatures with semi-trusted adjudicator," *Comput. Elect. Eng.*, vol. 36, no. 6, pp. 1035–1045, 2010.
[29] M. Swan, *Blockchain: Blueprint for a New Economy*. Newton, MA, USA: O'Reilly Media, 2015. [Online]. Available: http://shop.oreilly.com/product/0636920037040.do
[30] *The European Parliament and the Council of the Europena Union: Regulation (EU) 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*, document 32014R0910, 2014.
[31] J. Zhou, R. Deng, and F. Bao, "Some remarks on a fair exchange protocol," in *Proc. 3rd Int. Workshop Pract. Theory Public Key Cryptosyst. (PKC)*, in Lecture Notes in Computer Science, vol. 1751. New York, NY, USA: Springer-Verlag, Jan. 2000, pp. 46–57.
[32] J. Zhou, J. A. Onieva, and J. Lopez, "Optimized multi-party certified email protocols," *Inf. Manage. Comput. Secur. J.*, vol. 13, no. 5, pp. 350–366, 2005.
[33] E. B. Barker and J. M. Kelsey, "Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST Special Publication 800-131A, Mar. 2019. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-131Ar2
[34] *Digital Signature Standard (DSS)*, Standard FIPS 186-4, National Institute of Standards and Technology, Washington, DC, USA, Jul. 2013. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.186-4
[35] *Secure Hash Standard (SHS)*, Standard FIPS 180-4, National Institute of Standards and Technology, Washington, DC, USA, Aug. 2015. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.180-4
[36] Standards for Efficient Cryptography. (2010). *SEC 2: Recommended Elliptic Curve Domain Parameters. Version 2.0*. [Online]. Available: https://www.secg.org

M. MAGDALENA PAYERAS-CAPELLÁ was born in Mallorca, Spain, in 1973. She received the Telecommunication Engineering degree from the Polytechnic University of Catalonia (UPC), in 1998, and the Ph.D. degree in computer science from the University of the Balearic Islands (UIB), in 2005.

She has held several positions at the Department of Mathematics and Computer Science, University of the Balearic Islands, since 1998, where she is currently an Associate Professor and the Head of the Telecommunications Engineer Master. She has taught subjects in grade courses and in several masters. She has directed a Ph.D. Thesis, in 2013, and is currently directing another one. She is the author of five book chapters and the coauthor of two patents. Her Ph.D. dissertation was awarded with the COIT, Engineers Spanish Association, best Ph.D. thesis, in 2005. She has participated in two European-funded research projects and five Spanish-funded research projects (including a Consolider–Ingenio Project) having been the PI of two of them. She is member of the research group SECOM and the ARES Consolider Excellence Network. She has participated in the creation of two spin-offs: the company KEYRON, established, in 2006, and the company GEDOCU IT Consulting established, in 2016. Author of more than ten articles published in international journals included in Journal Citation Reports (JCR) in the last decade. Her research is focused primarily in security in communications networks, electronic payments, design of protocols for electronic commerce and secure applications, privacy-preserving applications, electronic ticketing, uses of cryptography, traffic management protocols, and technical-legal aspects of electronic commerce. Within these lines of research, she has had an active pace of publications in international journals and international and national conferences.

Dr. Capellá has been a member of the scientific and organizing committee in several international congresses and has wined three best paper awards for papers presented in conferences.

MACIÀ MUT-PUIGSERVER was born in Mallorca, Spain, in 1966. He graduated in computer science from the Autonomous University of Barcelona, in 1990. He received the Ph.D. degree from the University of Balearic Islands (UIB), in 2006.

In 1996, he joined the Department of Mathematics and Computer Science, UIB, as an Associate Lecturer, where he has taught subjects in different grade courses and masters. As a Researcher, he has been involved in European-funded research projects and some Spanish-funded research projects. He is currently a member of the Security and Electronic Commerce (SECOM) research group, UIB, and the ARES Consolider Excellence Network, a granted network in security and privacy with the objective of encouraging the cooperation between researchers. He has an active pace of publications on these topics in international journals, conferences, and book chapters. Most recent research published is focused on the incorporation of blockchain technologies in secure and e-commerce applications. His current research interests include the design of secure protocols, secure e-commerce, mobile applications, privacy-preserving applications, and applied cryptography.

MIQUEL A. CABOT-NADAL was born in Mallorca, Spain, in 1975. He received the Computer Engineering degree from the Open University of Catalonia (UOC), in 2009, and the master's degree in information and communications technologies (ICT) from the University of the Balearic Islands (UIB), in 2011, where he is currently pursuing the Ph.D. degree in information and communications technologies. He has a few conference and journal publications. His research interests include security, blockchain, and electronic commerce.