IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 30, 2019, accepted February 25, 2019, date of publication February 28, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902174

A Solution for Secure Certified Electronic Mail Using Blockchain as a Secure Message Board

M. FRANCISCA HINAREJOS, JOSEP-LLUIS FERRER-GOMILA, AND LLORENÇ HUGUET-ROTGER

Department of Computer Science, University of the Balearic Islands, 07122 Palma de Mallorca, Spain

Corresponding author: M. Francisca Hinarejos (xisca.hinarejos@uib.es)

This work was supported in part by the European Social Fund and in part by the Spanish Government under Project TIN2014-54945-R and Project TIN2015-70054-REDC.

ABSTRACT The certified mail is a value-added service that is widely used in the paper world. However, the scientific community has not yet provided a solution for certified e-mail that has achieved widespread acceptance. This lack of a certified e-mail solution is not due to a lack of proposed approaches; because over the past 40 years, more than 100 protocols have been reported in journals and at conferences. The vast majority of these proposed protocols use a trusted third party (TTP) to achieve fairness. The few solutions without a TTP have not been successful due to their high computational and/or communication cost. Blockchain provides a new approach to develop the protocols without a TTP but without the prior drawbacks of the previous solutions without a TTP. Here, we present a new protocol for certified e-mail based on a blockchain without a conventional TTP that is integrated with the conventional e-mail infrastructure. The protocol is secure, efficient, and viable from a practical perspective.

INDEX TERMS Bitcoin, blockchain, certified delivery, certified email, email security, fair exchange, nonrepudiation.

I. INTRODUCTION

Certified mail is a service provided by postal companies, and it is useful in multiple situations, such as providing notifications by public administrations. The sender of the mail wants to have evidence (that cannot be refuted) that the recipient has received the mail; therefore, the sender does not want the mail to be delivered without obtaining the recipient's acknowledgment of receipt. In this situation, we are faced with a problem of simultaneity: the recipient will not sign an acknowledgment of receipt if he does not receive the mail, but the mail should not be delivered if there is no guarantee that the recipient will provide a signed acknowledgment of receipt.

This problem may be solved with the intervention of a trusted third party (TTP): the postal agent. This agent must personally deliver (in a face-to-face exchange) the mail if and only if he obtains the acknowledgment of receipt in exchange. This system is functional and has been in operation for many years, although it is far from perfect from the perspective of security. The agent can be dishonest and allow the recipient to view the contents of the mail before requesting the acknowledgment of receipt, thereby allowing the selective rejection of mail (for example, fines). Conversely, on many occasions, the recipient signs an acknowledgment of receipt indicating that he has received an envelope from a certain sender but later finds that the envelope is empty or that it contains an erroneous document.

In the electronic world, a certified email service is also of interest, and the problem that must be solved is the same: the ''simultaneity'' of exchanging the mail with acknowledging the receipt of the mail. We immediately observe a difference between physical mail and electronic mail: in the electronic world, a face-to-face exchange between a postal agent and the recipient is not possible. However, it is possible to have the support of a TTP, which allows solving the problem of simultaneity, or as expressed in this area, ensuring that the exchange is fair.

In fact, over the past 40 years, many approaches have been proposed that depend on the existence and possible intervention of a TTP. In some of these proposed approaches, the TTP intervenes in all exchanges (solutions with an

The associate editor coordinating the review of this manuscript and approving it for publication was Gaurav Somani.

online TTP), and in others, the TTP only intervenes in cases of conflict (solutions with an offline TTP). This type of solution typically receives two types of criticism. First, the TTP can become a bottleneck. Second, it can be difficult for the parties to agree on a TTP that is trusted by both parties.

Because of previous criticisms, a smaller set of proposed approaches present solutions in which the authors propose not using TTPs. There are different types of solutions without TTPs, but they are also not free of problems. Some solutions are based on the gradual exchange of information, others are based on probabilistic schemes, and a third group uses elements that behave as TTPs (although the authors do not clearly indicate it). The most common criticism is that real solutions without a TTP involve a high computational and/or communication overhead.

We propose a completely different solution that is based on a blockchain. Some authors have proposed blockchain-based solutions for a similar problem: contract signing. However, to the best of our knowledge, we are the first to propose using a blockchain for certified email, where the blockchain is integrated with the current email infrastructure. Our proposed approach is a solution without a TTP as the concept of TTP is currently understood. However, within our protocol, we consider the entities that compose the structure of a blockchain network to be third parties. Additionally, although none of these entities are trustworthy, we do understand that entities as a whole provide sufficient confidence for the solution to be secure, which means that the solution is fair.

Contributions. In this article, we provide a new solution for certified email based on a blockchain without using conventional TTPs and using the current email standard. The security analysis shows that the proposed solution satisfies the necessary security requirements for this type of protocol: fairness, timeliness, nonrepudiation, and confidentiality. We choose Bitcoin as a useful blockchain solution to implement our proposal. In addition, we conduct a study of the usage cost of the proposed approach that demonstrates the viability of the presented solution.

Organization. This paper is organized as follows. In section II, we explain the background related to Bitcoin, showing its ability to store data. In section III, we analyze the related approaches found in the literature, and we define the security requirements for certified email. Section IV is devoted to fully describing the new protocol for certified email, which is followed by its security analysis in section V. In section VI, the usage cost of the proposed solution is analyzed. Finally, the conclusions are presented in section VII.

II. BACKGROUND

A. THE BITCOIN SYSTEM

Bitcoin is a cryptocurrency based on public key cryptography, and it was proposed in 2008 in a paper authored by someone using the pseudonym Nakamoto [1]. Bitcoin became fully functional in January 2009, and its broad adoption has led it to become the most successful virtual currency. The main property of Bitcoin is its untrusted environment and distributed consensus algorithm that allows multiple parties to agree on a particular state of the system without having to trust each other. The system state is stored in an append-only data structure called a blockchain.

Bitcoin is based on accounting entries. A Bitcoin account is defined by an elliptic curve cryptography key pair. A Bitcoin account is publicly identified by its Bitcoin address, which is obtained from its public key using a unidirectional function. Using this public information, users can send bitcoins to that address. Then, the corresponding private key is needed to spend the bitcoins in the account. Payments in the Bitcoin system are performed through transactions between Bitcoin accounts. A Bitcoin transaction indicates the movement of a bitcoin from a source address to a destination address. Each Bitcoin transaction has two main parts: inputs and outputs. Every input indicates an output from a previous transaction where bitcoins will be spent. Such output is identified by a transaction id (the hash of the transaction) and an output sequential value (since transactions may contain multiple outputs).[1] The outputs of the transaction indicate where the bitcoins of the inputs will be stored as new outputs. Outputs that have not already been spent are known as unspent transaction outputs (UTXO), and they represent the total amount of bitcoins in circulation. In addition to inputs and outputs, Bitcoin transactions also contain another field called `Locktime`, which indicates the earliest time that the transaction can be added to the blockchain. This time can be specified as an absolute time or as a block height.

The correctness of a Bitcoin transaction (also known as a payment) is based on two[2] main validations: proof of ownership and double spending.

Proof of ownership allows the transaction validator to verify that the sender of the transaction has the right to do so, that is, she is in possession of the bitcoins indicated in the input part of the transaction. The sender has to provide the conditions needed to unlock the bitcoins that appear in the previous outputs that the transaction is spending. In a regular transaction, such conditions include the digital signature of the sender performed with the private key linked to the address where the bitcoins come from.[3] The non-double-spending validation is performed over the set of all transactions previously processed by the Bitcoin system included in the blockchain. To validate the new transaction, all inputs of the transaction must point to a UTXO, that is, an output that has not been previously spent.

[1] With such a reference model, each output may only be in a binary state (spent or not spent), and it is not possible to spend just part of an output.

[2] Although multiple validations have to be performed over a transaction, these are the two main validations needed to understand our proposal. Interested readers may refer to https://goo.gl/fSWRk1 for a detailed list of transaction validations.

[3] This is a simplification of the system since more complex conditions may be needed to lock or unlock funds in a Bitcoin transaction. Interested readers may refer to [2, Ch. 5] for further details.

Note that to be able to control double spending, the system needs a ledger in which all previous transactions are annotated, and the integrity of the information included in the ledger must be preserved, although the ledger may allow adding new transactions. The blockchain is such an append-only ledger that contains all Bitcoin transactions performed since the system started operating in 2009. This ledger is freely replicated and stored on different nodes of the Bitcoin network, making Bitcoin a completely distributed system.

Transactions are included in the blockchain at time intervals rather than in a flow fashion, and this addition of transactions is performed by collecting all new transactions of the system, compiling them together in a data structure called blocks, and including the block at the top of the blockchain. Every time that a block that contains a specific transaction is included in the blockchain, the transaction is said to be a confirmed transaction since it has already been included in the blockchain and can be checked for preventing double spending.

Blocks are data structures that mainly contain a set of transactions that have been performed in the system. To achieve the append-only property, the addition of a block to the blockchain incurs a high computational cost; thus, adding blocks to the blockchain is time consuming and computationally intensive. Furthermore, every block is indexed using its hash value, and every new block contains the hash value of the previous block. This mechanism ensures that the modification of a block from the middle of the chain would imply modifying all remaining blocks of the chain from that point to the top to match all hash values.

Adding a block to the blockchain is known as the mining process; this process is also distributed and can be performed by any user of the Bitcoin network using specific-purpose software (and hardware). The mining process uses a hashcash proof-of-work system, which was first proposed by Adam Back as an antispam mechanism [3]. The proof-of-work consists of finding a hash of the new block with a value lower than a predefined target.[4] This process is performed by brute force, varying the nonce value of the block and hashing the block until the desired value is obtained. Once the value has been found, the new block becomes the top block of the blockchain, and all miners discard their work on that block and move to the next one by collecting new transactions and taking the hash of the top block as the previous block hash.

Mining new blocks is a structural task in the Bitcoin system since it helps to confirm the transactions of the system. For this reason and also assuming that mining implies hard work, miners have to be properly rewarded. In the Bitcoin system, miners are rewarded through two mechanisms. The first one provides them with newly created bitcoins. Every new block includes a special transaction, called a generation transaction, in which no input address appears and the output address is determined by the miner who creates the block, who clearly indicates one of their own addresses. The fee that each transaction pays to the miner is the second reward mechanism. The fee for each transaction is calculated by computing the difference between the total input amount and the total output amount of the transaction. All fees collected from transactions in a block are included in the generation transaction.

[4]The value of the target determines the difficulty of the mining process. The Bitcoin system adjusts the target value depending on the hash power of the miners to set the throughput of new blocks to 1 every 10 minutes (on average).

B. THE BITCOIN BLOCKCHAIN AS A SECURE MESSAGE BOARD

The ability to store information in the blockchain in addition to only the data related to the Bitcoin system was realized early, and different techniques were used for this purpose, from using the amount of a transaction as a data storage (with its limitations) to encoding information as Bitcoin addresses, which makes them unspendable since the corresponding private key is unknown. However, this mechanism conflicts with the standard use of the payment system since the outputs generated for those purposes are not distinguishable from standard payment outputs and wallets need to keep track of them even though they will never be able to be spent. To address such inefficiencies, the Bitcoin system defined a special-purpose output, the `OP_RETURN`, that could include a small amount of arbitrary data.[5] Such output is marked as nonspendable so wallets do not need to track them. Figure 1 shows a transaction used by Alice to post a message in the blockchain.

FIGURE 1. A high-level picture of a general bitcoin transaction, including an `OP_RETURN` output.

The interest in using a blockchain as a message board arises because its append-only property makes it suitable as a message board with strong security properties. On the one hand, the fact that blocks include their creation date allows using transactions as timestamped messages. On the other hand, the immutability of the blockchain provides the means for secure timestamping. Furthermore, in contrast to standard

[5]As of June 2018, with Bitcoin Core v0.16.1, the OP RETURN payload limit is set to 83 bytes. However, this limit has changed over time.

timestamping systems in which a timestamp authority has a central role in determining which information can be dated, the Bitcoin system avoids such a TTP role, allowing a distributed and sensor-resistant timestamping mechanism.

However, such a service does not come for free, and including a message in the Bitcoin blockchain has a price since every transaction in the Bitcoin network always pays a fee, irrespective of whether it carries a message (`OP_RETURN` output). The price of the message can be determined based on two parameters: the size of the message and the time that it takes to be published in the blockchain. On the one hand, since space in an immutable storage is a scarce resource, transaction fees are determined by their size in bytes, and the standard way to quantify the fee is based on a value indicating the ratio of satoshi/byte.[6] On the other hand, transactions are included in the blockchain in blocks, and such blocks appear, on average, every 10 minutes and have a 1 MB limit to include transactions. Transactions compete to enter in the next block with their fees. Miners tend to include transactions with greater fees first to increase their profits. Therefore, in a congestion period when users generate more transactions than the system can process, the time that it takes for a transaction (and its included message) to be published in the blockchain may be longer than 10 minutes, and it will be determined by the amount of fees that it carries.

Although we focus on Bitcoin, which is the best known and extended cryptocurrency platform, any blockchain with an operation similar to Bitcoin and that includes the `OP_RETURN` field can be used in our proposal. For example, Litecoin and Dash allow data publishing and use the same functionalities, structure and format as Bitcoin. Thus, although we will explain the proposal based on Bitcoin, any of these alternative blockchain platforms can be used. Throughout this article we refer to these alternatives as Bitcoin-based solutions.

III. RELATED WORK AND SECURITY REQUIREMENTS

Traditionally, solutions for certified email have been divided into two groups: without a TTP and with a TTP. Solutions without a TTP [4]–[7] do not require any external entity to help make the exchange fair. The first solutions of this type were based on the gradual release of secrets. Subsequently, probabilistic approaches were proposed. Solutions without a TTP have commonly been criticized because they involve a high computational and/or communication cost [8].

Regarding solutions with a TTP, these are subdivided into two large groups: with an online TTP and with an offline TTP. In the solutions with an online TTP (e.g., [9]–[13]), a TTP appears that intervenes in each exchange, and its intervention is necessary to guarantee fairness. This type of solution receives two criticisms: first, the TTP can become a bottleneck, and second, it can be difficult for the parties to agree on a TTP that gives them both confidence.

[6]A satoshi is a 10^{-8} part of a bitcoin.

To solve the bottleneck problem, solutions with an offline TTP or optimistic arise (e.g., [14]–[26]). In these solutions, the TTP only intervenes in cases of conflict and not in each execution. Although it is true that they can solve the bottleneck problem, assuming that the conflict rate will be low, the problem of the parties agreeing on a TTP remains.

The blockchain, particularly Bitcoin and derived cryptocurrencies, emerged as a different way to solve the security problem of double spending on electronic money. However, other researchers have proposed alternative uses, taking advantage of the fact that the created infrastructure provides a public and ''unalterable'' registry. This would solve the problem of the parties having to agree on a TTP. In addition, this property has been complemented with the appearance of smart contracts. Thus, we find in the literature a few proposed approaches for the electronic signature of contracts (e.g., [27]–[30]), payment by receipt (or product) (e.g., [31]–[33]), or proof of existence (e.g., [34]–[36]). However, we have not found any protocol for certified email using a blockchain.

Smart contracts are a useful tool when some operations must be controlled; for example, Bitcoin allows imposing restrictions and penalties using transaction scripts. This idea is used by some protocols to compensate the honest parties when a party leaves the protocol execution with an advantage over the remaining parties [30], [33], [37]–[39]. For example, the approach presented in [38] for a fair protocol for data trading uses locked transactions, where the seller cannot redeem the payment unless she provides a secret; this is the private key used by the buyer to decrypt the digital goods. Thus, at the time when the seller accesses the money, the buyer can access the product. Huang *et al.* [30] propose a three-party protocol for contract signing, where the definition of the output script of the transaction controls the conditions of the protocol, allowing a monetary penalty for a party aborting the protocol prematurely. Moreover, the protocol allows the privacy of the contract data to be signed because of the use of threshold public key encryption and verifiable encryption. The same authors describe an extension of this protocol applied to the fog computing scenario [40], where a three-layer service model appears: cloud servers, users and fog nodes.

Other sets of proposals use Bitcoin as a proof of existence platform [34]–[36]. In [34], a time-stamping service is presented. In this case, a service provider is responsible for publishing the proof of existence of a document. They use Bitcoin to publish a Merkle hash root, where each leaf of the Merkle tree depicts the hash of a document. Thus, they maintain the privacy of the document (only a hash is published, not revealing any data about it). However, a user cannot verify the existence of a document without interacting with the service provider; it must provide more information to verify that a document exists at a certain moment in time. The same concept is used in [35] to timestamp a two-party contract signature and in [36] to anchor and certify the existence and ownership of data.

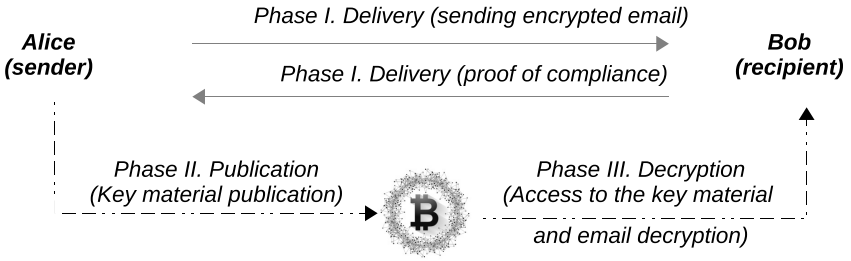

FIGURE 2. A sketch of the certified email proposal.

Other fair exchange proposals use Bitcoin as a fair payment scheme, avoiding the involvement of a TTP to the greatest extent possible. Jayasinghe *et al.* [31] present an optimistic fair exchange where a TTP and Bitcoin are involved, but Bitcoin is used just to pay for the product from a customer to a merchant in an e-commerce transaction, when the customer has all the proofs to obtain the product (with or without intervention of the TTP). When the merchant receives the bitcoin payment, he sends the product decryption key to the customer, and she can obtain the original product. If she does not receive the correct key, then she can make contact with the TTP to resolve this situation. Liu *et al.* [32] present two proposals for payment-for-receipt using smart contracts based on signatures to provide strong timeliness and based on fixed time-outs achieving weak timeliness. In both schemes, a TTP intervenes, but the smart contract is used to maintain the state of the protocol, alleviating this burden on the TTP. Zhang *et al.* [33] design the BCPay payment framework based on the scheme proposed by Andrychowicz *et al.* [27], but in this case, the scheme allows for fair payment for outsourcing services in cloud computing.

Finally, in relation to commercial solutions, we have found only two references that are somewhat related to our objective. On the one hand, we have ''Invisible Ink'', an MIT project that, according to its website, began and ended in January 2015, but no specifications are available. On the other hand, we have the SwiftMail application of McAfee [41]. The protocol specifications (designed and implemented) are not available. The fact that no details of the protocol are provided, in particular, the use of the blockchain, prevents us from conducting an analysis of its security. From the information available, we can conclude that, although SwiftMail is an interesting tool, it departs from one of our objectives: that the certified email service can be integrated into conventional email without having to make changes to the transport infrastructure of the emails. SwiftMail is a certified delivery service that does not use (or integrate with) SMTP-based Internet email.

As a conclusion, we do not find solutions for certified email. Moreover, we want to provide a solution that is as simple as possible, requiring the use of the blockchain only when required. Thus, we propose a certified email solution that avoids the use of smart contracts or the involvement of a third party to provide the service of data publishing in the blockchain.

A. SECURITY REQUIREMENTS

Regarding the security requirements, it must be kept in mind that certified email protocols are a subset of the family of fair exchange protocols. A list of security requirements for certified email can be found in [42], where the authors provide a critical review. Next, we provide definitions of the security requirements that apply to our scenario, in which there is no conventional TTP:

Fairness ''A certified email protocol fulfills the fairness property if and only if by the end of the execution, a) both the sender and recipient receive the expected items (or will receive them in a finite amount of time) or b) neither of them receives what it expects''.

Timeliness ''A certified email protocol fulfills timeliness if and only if honest entities can unilaterally choose to terminate the protocol in a finite amount of time without losing fairness''.

Confidentiality ''A certified email protocol fulfills the confidentiality property if and only if for the correct and fair protocol execution, no third entities need access to the message content''.

Nonrepudiation of Origin (NRO) ''A protocol provides a nonrepudiation of origin service if and only if it generates evidence of origin that will allow the recipient to demonstrate to an arbiter whether the originator was or was not the message's author''.

Nonrepudiation of Receipt (NRR) ''A protocol provides a nonrepudiation of receipt service if and only if it generates evidence of receipt that will allow the originator to demonstrate to an arbiter whether the recipient received the message''.

IV. A CERTIFIED EMAIL PROTOCOL

A. PROPOSAL SKETCH AND ASSUMPTIONS

In this section, we overview our certified email protocol based on Bitcoin (as the reference implementation) to provide fairness between the sender (*Alice*) and the recipient (*Bob*) without the involvement of a centralized TTP. Figure 2 depicts the relationships and the main steps followed by the involved participants. All of these steps can be grouped into

three phases. In the first phase, *Alice* sends the encrypted email to *Bob* along with a proof of delivery (a signature), which is one of the parts of the key material required to decrypt the certified email, and a time mark indicating a time limit for the complete delivery of the certified email. Clearly, *Alice* knows *Bob*'s email address. Once *Bob* receives these data, he sends a proof of compliance of the email delivery. Upon receiving this proof, *Alice* starts the second phase, publishing the second part of the key material in the blockchain. This key material should be published before the time mark specified in the first phase. After this time, *Bob* can begin the third phase to obtain the key material published by *Alice* in the blockchain. With all the key material, *Bob* can derive the decryption key and decrypt the certified email. Remember that blockchain denotes the structure of blocks where the data are recorded, as specified in the background section.

All the generated proofs are signed by *Alice* or *Bob*; therefore, they need a key pair (public-private), and the public key must be trusted by the other party. Moreover, we also assume that *Alice* and *Bob* use a resilient channel, that is, messages can be delayed but they will arrive in a finite amount of time. To address this issue, a time mark (a deadline) will be used to ensure timeliness, and in some ways, fairness.

Since the protocol uses Bitcoin, at least *Alice* needs to be connected to the Bitcoin network to send transactions conveying the required data (key material). Thus, *Alice* can be considered an active user of Bitcoin, requiring the use of a Bitcoin public address. Unlike *Alice*, *Bob* only needs to scan the network for the required key material published by *Alice*; thus, he is not required to send or receive transactions. In this sense, *Bob* can be considered a passive user of Bitcoin.

B. PROTOCOL SPECIFICATIONS

As explained above, our certified email protocol can be divided in three main phases (Phase I *Delivery*, Phase II *Blockchain Publication* and Phase III *Decryption*), which we next describe in detail. Table 1 shows the notations used along with the protocol description.

TABLE 1. Certified email protocol parameters.

M	email content
NRO	Nonrepudiation of origin
NRR	Nonrepudiation of receipt
k	master key
k_1	derivation key
k_2	decryption key
$E_k()$	Encryption with the symmetric master key k
$D_k()$	Decryption with the symmetric master key k
t_d	time delivery deadline
t_{nrr}	time when $Alice$ receives NRR_1 from Bob
$h()$	one-way collision-resistant hash function
$Sig_Y(x)$	signature on the element x made by party Y
$PU_B(k_2)$	k_2 encrypted with Bob's public key
Adr_X	Bitcoin address of party X

Phase I - Delivery: During Phase I, *Alice* and *Bob* exchange the main evidence of the email protocol (see Fig. 3), the nonrepudiation of origin (*NRO*) and the first part of the nonrepudiation of receipt (NRR_1), as explained below.

Step 1. $A \rightarrow B$: *encrypted email*
$m_1 = \{cem, NRO\}$
where: $cem = \{C, t_d, K^{'}, Adr_A\}$
$C = E_k(M)$
$K^{'} = PU_B(k_2)$
$NRO = Sig_A(cem)$

Step 2. $B \rightarrow A$: *proof of compliance*
$m_2 = \{NRR_1\}$
where: $NRR_1 = Sig_B(cem)$

FIGURE 3. Phase I- Email Delivery. Messages between *Alice* and *Bob*.

In Step 1, *Alice* sends what can be considered a token of commitment *cem*, without risking herself to lose fairness. Before creating the token of commitment, *Alice* generates the following key components for this email delivery:

- *master key* k: a random value that represents the secret key used to encrypt the email content (M).
- *derivation key* k_1: a random value that has the same length as the secret key.
- *decryption key* k_2: a value that is computed from k_1 and k as follows:

$$k_2 = k \oplus k_1$$

Once generated, the commitment (*cem*) is composed with the following values:

- The ciphertext C, that is, the message M encrypted with the *master key* k known only by *Alice* (*Bob* will not be able to read the message M until a later phase in the protocol).
- A deadline time t_d to which *Alice* commits to have published the key k_1 that allows *Bob* to decrypt the ciphertext C. It is useful to achieve timeliness. In fact, *Alice* assumes the following commitments if she receives the acknowledgment from *Bob* (NRR_1):
  1) she will publish the *derivation key* (k_1) in the blockchain before t_d, and
  2) if the transaction (containing the key) is not included in the blockchain before t_d, the acknowledgment of receipt that *Bob* may have sent will be without effect.
- k_2 is encrypted with the public key of *Bob*, $K' = PU_B(k_2)$, to provide confidentiality to the exchange.
- A Bitcoin address Adr_A from which *A* knows the corresponding private key.

Finally, *Alice* must sign all previous information in a nonrepudiation of origin token, $NRO = Sig_A(cem)$, and therefore, *Alice* will not be able to deny having sent it.

In Step 2, upon receiving m_1, if *Bob* does not want to receive the certified email, he should simply ignore the message received from *Alice*. If he wants to receive the email, he must send a message (see *Step 2* in Fig. 3) with the following content:

- the signature of *Bob* on the token *cem* (received from *Alice*): $NRR_1 = Sig_B(cem)$. This signature is one of the parts of the nonrepudiation token (*NRR*) for *Alice*.

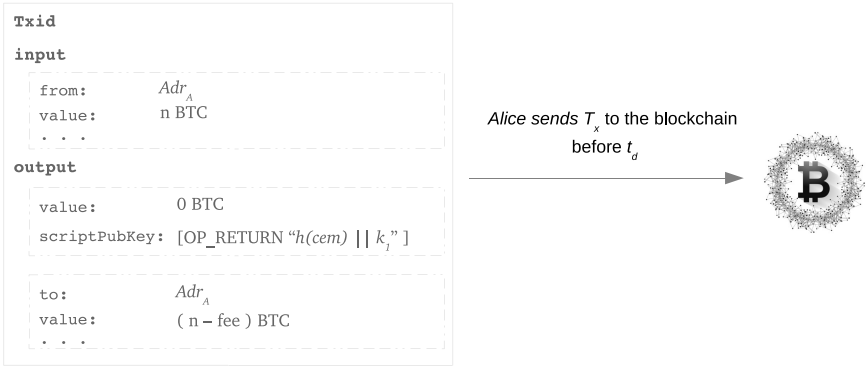

FIGURE 4. Phase II - blockchain publication.

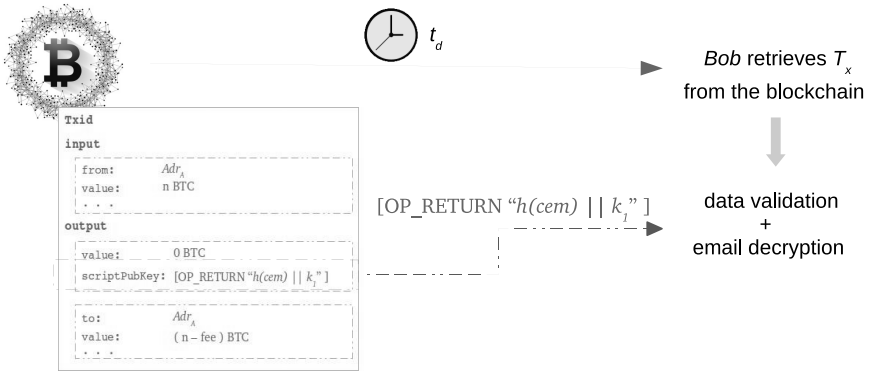

FIGURE 5. Phase III - bob retrieves tx_1 from the blockchain.

Phase II - Blockchain Publication: The NRR_1, which is sent by *Bob* to *Alice*, confirms that *Bob* wants to receive the certified email, but it is not useful for *Alice* if she does not achieve the derivation key (k_1) being published as a valid transaction tx_1 in the blockchain before t_d. Now, *Alice* should evaluate whether there is enough time to "guarantee" that if she sends the transaction tx_1 to the blockchain network, then it will be validated and published in the blockchain before t_d. This decision is based on the amount of fees that *Alice* is willing to pay (see Section VI for a cost analysis). If *Alice* decides to proceed, the following steps are performed:

- In Step 1, Alice creates a Bitcoin transaction tx with an `OP_RETURN` output that includes the value $h(cem) \parallel k_1$, where $\parallel$ denotes concatenation. The input of transaction tx_1 spends bitcoins from address Adr_A indicated in the *cem* data (see Fig. 4). Recall that `OP_RETURN` allows publishing up to 83 bytes (see Section II); thus, when using AES with a key of 256 bits and an SHA-256 hash function, only 64 bytes are required. Then, Alice broadcasts the transaction tx_1 to the Bitcoin network, allowing Bitcoin miners to obtain the transaction and include it in a mined block.
- In Step 2, the transaction tx_1 is published in the blockchain before t_d.

With the publication of tx_1 in the blockchain before t_d, *Alice* obtains the complete nonrepudiation of receipt evidence *NRR*, which is composed of two elements:

- NRR_1 received from *Bob*, and
- tx_1, the transaction validated and published in a block of the blockchain before t_d and conveying the derivation key k_1.

The publication of k_1 does not pose any risk to the security of the exchange (in particular, confidentiality), as explained in section V (Security Analysis).

Phase III - Decryption: Having the identifier of tx_1 (or the address Adr_A), *Bob* may retrieve the transaction from the blockchain to obtain the key to decrypt the email *cem*. For this purpose, *Bob* performs the following steps:

- Retrieve tx_1 from the blockchain (see Fig. 5).
- Validate that tx_1 includes an `OP_RETURN` output with the values $h(cem) \parallel k_1$.
- Validate that tx_1 has been correctly signed by *Alice*. Strictly, such validation is not necessary because the fact that tx_1 is included in the blockchain implies that the verification has already been performed by the Bitcoin network; in other words, the input of the transaction tx_1 corresponds to the address Adr_A.
- Validate that $h(cem)$ computed from *cem* received in Phase I - Step 1 is equal to $h(cem)$ extracted from tx_1.
- Retrieve k_1 and perform a bitwise XOR with the value k_2 retrieved from the *cem* message in Phase I to obtain the master secret k

$$k_2 \oplus k_1 = (k \oplus k_1) \oplus k_1 = k$$

and to be able to decrypt C, obtaining M

$$D_k(C) = D_k(E_k(M)) = M$$

V. SECURITY ANALYSIS

In this section, we will review the accomplishment of the requirements that we presented in Section III to prove they are met by our proposed protocol.

A. FAIRNESS

Here, we must prove that the protocol is fair for an honest party, irrespective of the actions performed by a dishonest party (*Alice* or *Bob*) or even an external attacker. We will analyze the possible attacks after each step of the protocol.

After Phase I, Step 1, *Alice* has sent *cem* to *Bob*, and no attack can be performed. *Bob* cannot read the message because he needs k_1, and *Alice* does not have the receipt from *Bob*. Therefore, this situation is fair for *Alice* and *Bob*. *Bob* may not send NRR_1, which is not a problem.

After Phase I, Step 2, *Bob* has sent NRR_1 to *Alice*, and no attack can be performed. Again, *Bob* cannot read the message because he does not have k_1. However, *Alice* does not have the complete receipt: k_1 must be validated and published in the blockchain. Therefore, this situation is fair for *Alice* and *Bob*. If *Alice* does not execute Phase II and does not publish k_1 in the blockchain, then *Bob* does not have the message, but *Alice* does not have the NRR, so it is a fair situation for both parties. In the case that *Bob* sends an invalid NRR_1, *Alice* must stop the exchange.

After Phase II, Step 1, *Alice* has broadcast the transaction tx_1 to the Bitcoin network, but tx_1 is still pending inclusion in the blockchain. At this point, we have three possible future situations with two possible outcomes.

- Case 1: tx_1 will be published in the blockchain before t_d.
- Case 2: tx_1 will never be published in the blockchain.
- Case 3: tx_1 will be published in the blockchain after t_d.

Case 1 is equivalent to Phase II - Step 2 that we will review later.

In cases 2 and 3, we will assume that *Bob* has obtained tx_1; otherwise, we will be in the same scenario after Phase I, Step 2, discussed previously. Note that in the event of either of these two cases, *Alice* will face an unfair situation since *Bob* will be able to read the message: he has k_2 from Phase I - Step 1, and he will be able to obtain k_1 from the knowledge of tx_1 and will therefore be able to decrypt *cem* and read M. By contrast, *Alice* will not obtain *NRR* since transaction tx_1 containing k_1 has not been published before t_d. However, in the next paragraphs, we will discuss how *Alice* can prevent such scenarios to avoid an unfair situation.

Case 2, where tx_1 is never published, implies that no miner has ever been aware of the existence of tx_1 since the incentive that the miners have in retrieving transaction fees implies that they are willing to include known transactions in blocks.[7] Regarding the broadcast mechanism of the information in the Bitcoin system, such possibility can only be derived from a delivered eclipse attack from *Bob* to *Alice*. An eclipse attack [43] is a well-known network attack in which *Bob* controls all of *Alice*'s Bitcoin network connections, thus allowing *Bob* to drop some/all transactions that will not reach the remainder of the Bitcoin network, and in particular, will not arrive to the miners and never be mined. Fortunately, different countermeasures have been implemented in the Bitcoin software client to avoid such attacks. Such measures are directed to randomize the peer discovery procedure used when a new peer accesses the Bitcoin P2P network and establishes connections with other peers to send and receive information from the system. Furthermore, *Alice* can establish a large number of peer connections, both from known sources and from random peers, to ensure reliable connectivity with the P2P network and avoid an eclipse attack.

In case 3, tx_1 is published in the blockchain after t_d. In this case, we will discard the possibility of an eclipse attack, as already discussed above, and we will focus on the possible delay in including the transaction in the blockchain due to the standard function of the Bitcoin system, described in Section II-B. This delay is related to the incentive that miners have to include such transactions in the blockchain, and such incentive is measured with the transaction fees. To avoid this situation, *Alice* measures the difference between t_d indicated in Step 1 of Phase I and t_{nrr}, the time that *Alice* has received NRR_1 in Step 2 of Phase I. This difference determines the time that *Alice* has for publishing the transaction in the blockchain. Then, based on the fees that *Alice* is willing to pay (see Section VI for more details), she can assess whether the transaction will be included before t_d in the blockchain and decide whether it is secure for her to broadcast the transaction to the Bitcoin P2P network, that is, whether she executes Step 1 of Phase II.

After Phase II, Step 2, when transaction tx_1 is published in the blockchain, no attack can be performed. If the publication of the transaction occurs before t_d, *Alice* has the *NRR* (NRR_1 and tx_1 included in the blockchain), and *Bob* can obtain k_1 from the blockchain and read the message M. Therefore, this situation is fair for *Alice* and *Bob*.

Note that at this point, if *Alice* acted maliciously and had published an invalid k_1 and/or sent and invalid k_2, then *Bob* will not be able to read M, but he will be able to prove it. The value of k_2 is signed by *Alice*, and k_1 is published in the blockchain in a transaction also signed by *Alice*. If *cem* cannot be decrypted with the master key, k, then *Bob* can prove it, and a judge can determine that *Bob* has not received message M.

As a conclusion, we can confirm that our proposal satisfies the fairness requirement.

B. TIMELINESS

The publishing deadline t_d establishes a time limit for the end of the exchange. The execution of the protocol can finish before that deadline (as soon as k_1 is published in

[7] Here, we assume that the Bitcoin system as a whole is fair. Even if some miners are dishonest and do not mine and discard received transactions, the incentive mechanism included in the Bitcoin system makes other miners behave as expected.

the blockchain), but at the latest, the parties will know the final state of the exchange when t_d is reached. Therefore, the timeliness requirement is met.

C. CONFIDENTIALITY

Alice sends the content of the certified message M encrypted with a symmetric key k (the master key). The master key is split into two parts: k_1 and k_2. k_1 is published in the blockchain and is accessible to everybody, but k_2 is encrypted with PU_B, the recipient's public key. Therefore, the content of the message is only known by the sender *Alice* and the recipient *Bob*, who is the only party capable of obtaining the master key k (adding k_1 and k_2).

Thus, we can confirm that the protocol meets the confidentiality requirement.

D. NONREPUDIATION

During the protocol execution, evidence is generated as nonrepudiation proofs for *Alice* and *Bob*. In particular, the protocol offers the following:

- Nonrepudiation of origin (*NRO*). *Alice* provides two signatures to *Bob*. In Step 1 of Phase 1, she signs *NRO*, which contains the ciphertext C (the content encrypted with the key k) and k_2 (encrypted with Bob's public key). Therefore, she cannot deny having sent that ciphertext and the value k_2. In Phase II, *Alice* signs the transaction tx_1 that contains the value k_1. Therefore, she cannot deny having published the value k_1 in the blockchain. The addition of k_2 and k_1 results in a key k that *Alice* cannot deny having sent. Consequently, she cannot deny having sent that information (ciphertext and key to decrypt it), thus meeting the nonrepudiation of origin requirement.
- Nonrepudiation of receipt (*NRR*). *Bob* provides one signature to *Alice*. In Step 2 of Phase I, he signs (NRR_1) the ciphertext C and the value k_2. Therefore, he cannot deny having received it. The protocol specification indicates that the publication of the value k_1 in the blockchain before t_d is the second part of the evidence for *Alice* (if *Bob* participates in the protocol execution, it means that he agrees with this condition). In short, if k_1 is published before t_d, then *Bob* cannot deny having received the encrypted message and the key to decrypt it, thus meeting the nonrepudiation of receipt requirement.

VI. USAGE COST OF THE PROPOSED APPROACH

As we previously mentioned, our proposed approach is a practical approach that can currently be implemented with the existing functionalities of the Bitcoin protocol, and it does not require any further modification of the Bitcoin specification. An issue that must be considered is the usage cost of our certified electronic mail solution.

To assess the usage cost of our protocol, we measure the added cost of the certification component, assuming that Internet email is already used, so no cost is measured for standard messages between both parties that can be performed by regular mail messages. With this assumption, the economic cost of our protocol is focused on the use of the Bitcoin network, which is restricted in Phase II of the description detailed in Section IV-B.

In Phase II, *Alice* creates a bitcoin transaction, in which she includes $h(cem)||k_1$ in the `OP_RETURN` field, and she publishes this transaction in the blockchain. As we already noted, including a transaction in the blockchain has a price: the transaction fee. The amount of this fee can be determined based on two parameters: the data size of the transaction and the time that it takes to be published in the blockchain. In the next subsections, we will analyze these parameters in detail.

A. THE SIZE OF THE DATA

Regarding the size of the transaction, we provide two different measurements. On the one hand, we suppose that *Alice* needs to create a completely new transaction to post the message in the blockchain. In this case, the fees of the transaction have to be considered as a cost for our protocol.[8] On the other hand, we assume that *Alice* is a regular user of the Bitcoin network, so she is performing regular bitcoin transactions that can be used to post the messages needed in our protocol. This assumption implies that the fees corresponding to the size of the transaction do not need to be assigned to our protocol since such payment transaction needs to pay them to be processed; thus, the cost that our protocol adds is the one corresponding only to the fees related with the size of the message that has to be included.

1) TOTAL TRANSACTION SIZE

To minimize the cost needed to pay for using our protocol, we will craft the smallest possible transaction that can carry the message. Such a transaction is a bitcoin transaction with one input and two outputs: one that transfers the bitcoins from the input to the original owner and a second one that carries the `OP_RETURN` with the inserted data. Furthermore, we will use the most compact address type, P2WPKH, for both input and output and also a Segwit transaction type, which further reduces the size needed in a block and thus reduces the fees. Recalling Section IV-B, the information that we need to include in the transaction is $h(cem)||k_1$, where $h(cem)$ is an SHA256 hash value and k_1 is an AES key of 256 bits, which accounts for the total amount of 64 bytes of data that needs to be included in the `OP_RETURN`.

With these parameters, our raw transaction has the following low-level structure:

[8] Only the fees are considered a cost since the value of the payment itself can move funds from two addresses belonging to *Alice* without having any economic impact on her.

Description	value (hex)	Size
version_field	02000000	4 bytes
Segwit flag	0001	2 bytes
Number of inputs	01	1 byte
Prev tx_id	variable	32 bytes
Prev output	variable	4 bytes
Script length	0	1 byte
Sequence	variable	4 bytes
Number of outputs	02	1 byte
import value	variable	8 bytes
P2WPKH script length	16	1 byte
P2WPKH script	0014+variable	22 bytes
import value	0000000000000000	8 bytes
OP_RETURN script length	42	1 byte
OP_RETURN code	6A	1 byte
Push data	40	1 byte
OP_RETURN data	variable	64 bytes
version byte	00	1 byte
Witness data length	47	1 byte
Witness data: signature	variable	71 bytes
Witness data length	21	1 byte
Witness data: public key	02+variable	33 bytes
LockTime	00000000	4 bytes

With these values, we are now able to compute the *virtual size* over which the fee is calculated (see [44] for more details). The virtual size is measured as follows:

$$vsize = \left\lceil \frac{btxs * 3 + ttxs}{4} \right\rceil$$

where *btxs* is the base transaction size, that is, the size of the transaction without Segwit data (Segwit flag and Witness data), and the total transaction size *ttxs* is the size of the complete transaction. Thus, the final virtual size in our proposed approach is:

$$vsize = \left\lceil \frac{(266 - 109) * 3 + 266}{4} \right\rceil = 185 \text{ bytes}$$

2) OP_RETURN only payload

The information that our protocol needs to include in the blockchain, 64 bytes, can be strictly specified using only an OP_RETURN output. The Bitcoin protocol establishes that every bitcoin transaction may carry a maximum of one OP_RETURN output. Thus, given a regular transaction that does not include an OP_RETURN, it is possible to add this type of output. Therefore, in the case that *Alice* uses an other-purpose transaction for including the data of our protocol, the cost for our protocol will be restricted to the added data in the transaction, that is, the OP_RETURN output part:

Description	value (hex)	Size
import value	0000000000000000	8 bytes
OP_RETURN script length	42	1 byte
OP_RETURN code	6A	1 byte
Push data	40	1 byte
OP_RETURN data	variable	64 bytes
Total		75 bytes

B. TIME FOR BLOCKCHAIN INCLUSION

Once we have determined the size of the data that have to be included in the blockchain, another relevant parameter to assess the usage cost of our protocol is the delay at which the transaction will be published once it has been broadcast in the Bitcoin P2P network. We will measure this delay in terms of blocks-to-publish. A transaction has delay 1 if it takes 1 block to be published or, in general, delay n when it takes n blocks to be published. This delay can be determined based on the fee of the transaction. Miners that find a transaction with a high fee have considerable incentive to mine it as soon as possible. Therefore, if the fee is high enough, the transaction delay will be the lowest possible, 1, which means that the transaction will be included in the next mined block. As far as the fee is reduced, miners will queue the transaction, and it will be mined with a high delay.

Measuring a transaction delay is a difficult task since multiple factors, some of them not publicly available, participate in the decision of a miner to include a transaction in a block. However, since such delay is a value needed for different purposes, existing wallets apply different techniques to estimate this value to provide the user with some degree of confidence on the delay a transaction has for appearing in the blockchain. In this paper, we will use the standard approximation that takes into account the difference between the time between a transaction is seen in the network and when it is included in a block and the fees that such a transaction carries. With such values, a fair estimation can be performed.

At present, the current fee estimation to include a transaction in the next block is[9] 42 satoshis per byte. With this fee and the current bitcoin value,[10] the usage cost of our protocol is \$0.28 for a complete transaction fee (that needs 185 bytes of space) or \$0.11 in case where an existing transaction is used and only the OP_RETURN payload (75 bytes) has to be paid.

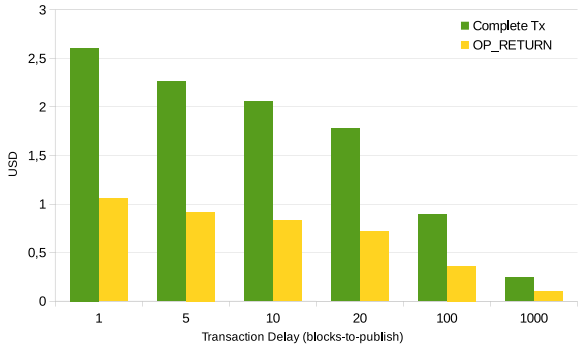

FIGURE 6. Average cost (in USD) depending on the publication delay of our protocol when a complete transaction is used.

In addition to this instant value, we have also measured the average usage cost of our proposed approach for a large period in which both fees and Bitcoin prices have been changing. In particular, we provide data[11] for the past year period,

[9]Date and time: 2018-11-26 / 23:59 - Source: https://statoshi.info/dashboard/db/fee-estimates.

[10]1 BTC = \$3640.56

[11]Historical data from Bitcoin fee estimation has been obtained from https://statoshi.info/dashboard/db/fee-estimates, and historical data of exchange rate between Bitcoin and dollars from https://coinmetrics.io/data-downloads/.

from 01/11/2017 to 31/10/2018. Figure 6 compares the usage cost when a completely new transaction is needed to publish the information in the blockchain versus the case in which only the OP_RETURN overhead should be assumed as a cost for different transaction delays.

As stated in the introduction section, alternatives to Bitcoin could be used to implement our proposal. Next, we analyze the cost required to deploy our proposal using some of the Bitcoin-based alternatives, such as Litecoin and Bitcoin Gold. Notably, both alternatives use the same OP_RETRURN field and segwit transaction as Bitcoin. Considering the current price of these cryptocurrencies[12] in US dollars, the cost of our proposal using Litecoin and Bitcoin Gold is $0.00699 and $0.00073, respectively. Therefore, the cost of using different Bitcoin-based solution can be greatly reduced.

In summary, the cost-time balance of the OP_RETURN payload publication is viable for the certified electronic mail scenario, where the average cost to publish in the next block (10 minutes) is $1.058, and the average cost to publish in the next 100 blocks (16 hours) is $0.3605 when using Bitcoin as the blockchain platform. Considering that the most economical solution identified in the business model requires a delay of 2-3 business days and implies a cost of $4.72 when requiring an electronic delivery confirmation plus $1.50 if additional options are required (such as a return receipt signature),[13] our proposed approach outperforms such solutions.

VII. CONCLUSIONS

In this article, we have presented a protocol for certified email based on the blockchain and using Bitcoin as a reference implementation model. Our proposed approach can be used without intermediate entities to satisfy the typical security requirements pursued in certified email (i.e., the approach does not require a TTP) or to publish the required key material on the blockchain. Meanwhile, the use of smart contracts, which can be expensive from both economical and temporal perspectives, is not necessary. Moreover, our solution is based on the current email infrastructure, and no changes to the email structure, known and certified addresses, or the certification infrastructure already used by people and businesses are required. In addition, we have analyzed the cost-time balance and shown that our proposed approach can be deployed with the current Bitcoin-based functionalities.

ACKNOWLEDGMENT

The authors would like thank J. Herrera-Joancomartí, researcher at the Department of Information Engineering and Communications of the Universitat Autònoma de Barcelona (UAB), for his advice and suggestions about Bitcoin.

[12]Date: 2016-11-21 Source: Open source cryptoasset analytics - url= "https://coinmetrics.io/"

[13]USPS Certified Mail Rates 2018 - https://www.certifiedmaillabels.com/usps-postal-rates

REFERENCES

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] A. M. Antonopoulos, *Mastering Bitcoin*. Newton, MA, USA: O'Reilly Media, 2014.

[3] A. Back, "Hashcash—A denial of service counter-measure," Tech. Rep., 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=3C5195E36D4548D1250C0F4AFF7AF07D?doi=10.1.1.15.8

[4] K. Shimizu, S. Miyazaki, and Y. Okabe, "Design and implementation of a certified mail exchange system using simultaneous secret exchange," in *Proc. 9th Annu. Int. Symp. Appl. Internet*, Jul. 2009, pp. 37–42.

[5] Y. Han, "Investigation of non-repudiation protocols," in *Information Security and Privacy. ACISP* (Lecture Notes in Computer Science), vol. 1172. Berlin, Germany: Springer, 1996, pp. 38–47.

[6] S. Ishibashi, S. Miyazaki, and Y. Okabe, "Design and implementation of a certified document delivery system without a trusted intermediate authority," in *Proc. IEEE/IPSJ 11th Int. Symp. Appl. Internet*, Jul. 2011, pp. 20–26.

[7] A. Paulin and T. Welzer, "A universal system for fair non-repudiable certified e-mail without a trusted third party," *Comput. Secur.*, vol. 32, pp. 207–218, Feb. 2013.

[8] H. Pagnia and F. C. Gärtner, "On the impossibility of fair exchange without a trusted third party," Darmstadt Univ. Technol.-Karolinenplatz, Darmstadt, Germany, Tech. Rep. TUD-BS-1999-02, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.7863

[9] M. Abadi, N. Glew, B. Horne, and B. Pinkas, "Certified email with a light on-line trusted third party: Design and implementation," in *Proc. Int. World Wide Web Conf. WWW*, 2002, pp. 387–395.

[10] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang, "Practical protocols for certified electronic mail," *J. Netw. Syst. Manage.*, vol. 4, no. 3, pp. 279–297, 1996.

[11] Y. Kavurucu and Ö. Sever, "Hybrid non-repudiation protocol with all types of pairings," *J. Naval Sci. Eng.*, vol. 12, no. 1, pp. 33–50, 2016.

[12] B. Schneier and J. Riordan, "A certified e-mail protocol," in *Proc. 14th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Washington, DC, USA, Dec. 1998, pp. 347–352.

[13] C.-H. Wang and C.-M. Wang, "Novel design of fair exchange protocol for semi-trusted server and its application in cloud environment," in *Proc. 11th Asia Joint Conf. Inf. Secur. (AsiaJCIS)*, Aug. 2016, pp. 130–135.

[14] G. Ateniese, B. de Medeiros, and M. T. Goodrich, "TRICERT: A distributed certified E-mail scheme," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2001, pp. 47–58.

[15] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for fair exchange," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 1997, pp. 7–17.

[16] G. Draper-Gil, J. L. Ferrer-Gomila, M. F. Hinarejos, and A. Tauber, "An optimistic certified e-mail protocol for the current Internet E-mail architecture," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2014, pp. 382–390.

[17] J.-L. Ferrer-Gomila, M. F. Hinarejos, G. Draper-Gil, and L. H. I. Rotger, "Optimistic protocol for certified electronic mail with verifiable TTP," *Comput. Standards Interfaces*, vol. 57, pp. 20–30, Mar. 2018.

[18] J. L. Ferrer-Gomila, M. Payeras-Capella, and L. H. I. Rotger, "An efficient protocol for certified electronic mail," in *Proc. Int. Workshop Inf. Secur. (ISW)*. London, U.K.: Springer-Verlag, 2000, pp. 237–248.

[19] S. Kremer and O. Markowitch, "Optimistic non-repudiable information exchange," in *Proc. Symp. Inf. Theory Benelux*, 2000, pp. 139–146.

[20] S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," *Comput. Commun.*, vol. 25, no. 17, pp. 1606–1621, 2002.

[21] Z. Liu, J. Pang, and C. Zhang, "Design and formal verification of a CEM protocol with transparent TTP," *Frontiers Comput. Sci.*, vol. 7, no. 2, pp. 279–297, 2013.

[22] S. Micali, "Simple and fast optimistic protocols for fair electronic exchange," in *Proc. ACM Symp. Princ. Distrib. Comput. (PODC)*, 2003, pp. 12–19.

[23] K. Maw and E. Khin, "A fair certified email protocol with message confidentiality," in *Proc. Int. Conf. Adv. Eng. Technol. (ICAET)*, Singapore, 2014, pp. 172–175.

[24] K. Maw and E. Khin, "Security enhancement of fair certified message delivery system," *Int. J. Sci. Eng. Technol. Res.*, vol. 3, no. 39, pp. 890–895, 2014.

[25] M.-H. Shao, G. Wang, and J. Zhou, "Some common attacks against certified email protocols and the countermeasures," *Comput. Commun.*, vol. 29, no. 15, pp. 2759–2769, 2006.
[26] G. Wang, "Generic non-repudiation protocols supporting transparent off-line TTP," *J. Comput. Secur.*, vol. 14, no. 5, pp. 441–467, 2006.
[27] M. Andrychowicz, S. Dziembowski, D. Malinowski, and Ł. Mazurek, "Fair two-party computations via bitcoin deposits," in *Financial Cryptography and Data Security. FC* (Lecture Notes in Computer Science), vol. 8438. Berlin, Germany: Springer, 2014, pp. 105–121.
[28] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in *Financial Cryptography and Data Security. FC* (Lecture Notes in Computer Science), vol. 9604. Berlin, Germany: Springer, 2016, pp. 43–60.
[29] H. Tian, J. He, and L. Fu, "Contract coin: Toward practical contract signing on blockchain," in *Information Security Practice and Experience. ISPEC* (Lecture Notes in Computer Science), vol. 10701. Cham, Switzerland: Springer, 2017, pp. 43–61.
[30] H. Huang, K.-C. Li, and X. Chen, "A fair three-party contract singing protocol based on blockchain," in *Cyberspace Safety and Security. CSS* (Lecture Notes in Computer Science), vol. 10581. Cham, Switzerland: Springer, 2017, pp. 72–85.
[31] D. Jayasinghe, K. Markantonakis, and K. Mayes, "Optimistic fair-exchange with anonymity for bitcoin users," in *Proc. IEEE 11th Int. Conf. E-Bus. Eng. (ICEBE)*, Nov. 2014, pp. 44–51.
[32] J. Liu, W. Li, G. O. Karame, and N. Asokan, "Toward fairness of cryptocurrency payments," *IEEE Security Privacy*, vol. 16, no. 3, pp. 81–89, May/Jun. 2018.
[33] Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, "Blockchain based efficient and robust fair payment for outsourcing services in cloud computing," *Inf. Sci.*, vol. 462, pp. 262–277, Jun. 2018.
[34] B. Gipp, N. Meuschke, and A. Gernandt. (2015). "Decentralized trusted timestamping using the crypto currency bitcoin." [Online]. Available: https://arxiv.org/abs/1502.04015
[35] Z. Wan, R. H. Deng, and D. Lee, "Electronic contract signing without using trusted third party," in *Network and System Security. NSS* (Lecture Notes in Computer Science), vol. 9408. Cham, Switzerland: Springer, 2015, pp. 386–394.
[36] A. S. de Pedro Crespo and L. I. C. García. (2017). "Stampery blockchain timestamping architecture (BTA)—Version 6." [Online]. Available: https://arxiv.org/abs/1711.04709
[37] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 8617. Berlin, Germany: Springer, 2014, pp. 421–439.
[38] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on Bitcoin transactions," *Future Gener. Comput. Syst.*, to be published. doi: 10.1016/j.future.2017.08.021.
[39] W. Zhang *et al.*, "Ttp-free fair exchange of digital signatures with bitcoin," in *Information Security Practice and Experience. ISPEC* (Lecture Notes in Computer Science), vol. 10701. Cham, Switzerland: Springer, 2017, pp. 62–81.
[40] H. Huang, K.-C. Li, and X. Chen, "Blockchain-based fair three-party contract signing protocol for fog computing," *Concurrency Comput., Pract. Exper.*, to be published. doi: 10.1002/cpe.4469.
[41] J. McAfee. (2015). *Swiftmail*. [Online]. Available: https://johnmcafeeswiftmail.com/
[42] 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.
[43] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on Bitcoin's peer-to-peer network," in *Proc. 24th USENIX Secur. Symp. (USENIX)*. Washington, DC, USA, 2015, pp. 129–144. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman
[44] E. Lombrozo, J. Lau, and P. Wuille. (2015). *BIP141: Segregated Witness (Consensus Layer)*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

M. FRANCISCA HINAREJOS received the M.S. degree in telecommunication engineering and the Ph.D. degree in computer science from the Technical University of Catalonia, in 2003 and 2010, respectively. Since 2014, she has been an Associate Professor with the Department of Mathematics and Computer Science, University of the Balearic Islands, Spain. She has authored several articles that are published in national and international conferences and journals. Her research interests include network security, electronic commerce, security in constrained environments, cybersecurity, and blockchain.

JOSEP-LLUIS FERRER-GOMILA received the M.S. degree in telecommunications engineering from the Technical University of Catalonia, in 1991, and the Ph.D. degree in computer science from the University of the Balearic Islands, in 1998, where he is currently an Assistant Professor with the Computer Science Department. He has authored several articles published in national and international conferences and journals. His research interests include network security and electronic commerce. He has been leading some national projects in this area.

LLORENÇ HUGUET-ROTGER received the M.S. degree in mathematics and the Ph.D. degree in computer science from the Universidad Autònoma de Barcelona, in 1977 and 1981, respectively. He is currently a Full Professor with the Computer Science Department, University of the Balearic Islands. He has authored more than 100 articles published in national and international conferences and international journals. His research interests include network security and coding theory.