Electronic Commerce Research and Applications 36 (2019) 100869

Contents lists available at ScienceDirect

Electronic Commerce Research and Applications

journal homepage: www.elsevier.com/locate/elerap

A fair contract signing protocol with blockchain support

Josep-Lluis Ferrer-Gomila, M. Francisca Hinarejos*, Andreu-Pere Isern-Deyà

Department of Mathematics and Computer Science, University of the Balearic Islands (UIB), Cra. de Valldemossa, km 7.5, Palma, Illes Balears, Spain

ARTICLE INFO

Keywords:
Fair exchange
Contract signing
Blockchain
Bitcoin
Ethereum
Cryptocurrency
Fintech
Digitalization
Digital economy
Trust-free system
E-commerce
Trust

ABSTRACT

Electronically signing contracts is fundamental for e-commerce transactions. The main property that contract signing protocols must achieve is fairness of the exchange. The solutions presented to date are divided into two major types: those that have a trusted third party (TTP) to achieve fairness and those that do not. In the literature, we find more than 40 published proposals, but none of these proposed protocols has become a recognized or de facto standard in the market. Blockchain has provided a new way to address classic problems such as double spending, as well as problems such as fairness. In this article, we present a protocol for contract signing based on blockchain. Our proposed protocol does not require a conventional TTP, and it does not present the disadvantages of solutions without a TTP (computational and/or communication cost). The protocol satisfies the necessary security requirements: fairness, timeliness and non-repudiation. We demonstrate the feasibility of the protocol with a cost analysis and a proof of concept implementation. In addition, we show how Ethereum can be integrated in our solution as an alternative platform to the use of Bitcoin. Finally, we show how our proposal improves previous solutions for contract signing based on blockchain in terms of cost, efficiency and security.

1. Introduction

Signing contracts is an essential process in the business of conventional commerce. Following a previous negotiation and agreement process, the next step, which is signing the contracts, can be performed remotely. This process includes a signer (*Alice*) who signs copies of the contract and then sends them to the other signer (*Bob*) so that he can sign the two copies and return one copy to *Alice*. Note that *Alice* takes the risk that *Bob* may not be honest and may be malicious when deciding whether to sign the contract (leaving *Alice* in an unfair situation). Therefore, when contracts require a certain level of security, the contracts are signed face to face. In certain situations, the contracts are signed in the presence of a trusted third party (TTP) (for example, a notary) to provide greater security to the signing of the contract.

Analogously, in the electronic world, protocols are necessary to sign contracts in a secure manner. The main problem that must be addressed is the same as in the physical world: the process has to be fair, that is, neither party is in an advantageous position. Regarding the solutions, we do observe a fundamental difference between the electronic and physical worlds: it is not possible to sign contracts face to face with an immediate exchange of signed copies. However, it is possible to have the collaboration of TTPs, in this case ”electronic”, to help solve the problem of fairness in exchanging the signed copies of the contract.

In the literature, we find two types of solutions for contract signing. The first type of solution uses the services of a TTP to ensure that the exchange is fair (e.g., Garay et al. (1999)). In turn, this type of solution is subdivided into two types. In the first one, the TTP intervenes in all exchanges (with an online TTP). In the second, the TTP only intervenes in the case of a conflict (with an offline TTP). These types of solutions are criticized because the TTP can become a bottleneck (especially if the solution is with an online TTP), and it can be difficult for the signers to agree on a TTP that is trusted by both parties.

The second main type of solution attempts to achieve the fairness requirement without the intervention of a TTP (e.g., Damgård (1995)). This type of solution has proven to not be viable from a practical perspective for several reasons: some solutions assume that the parties have the same computing power, and other solutions involve a high cost of computing and/or communications.

We agree with Hawlitschek et al. (2018) in that “overcoming the trust frontier without the necessity of trusted third parties will be a major challenge for future work, and that ”blockchain technology is to some degree suitable to replace trust in platform providers”.

In line with the above, and to arrive at our proposed protocol, we start from the hypothesis that blockchain could allow the fairness requirement to be achieved through an approach that is different from the classic ones followed to date. It is true that the best known use of the blockchain is the development of cryptocurrencies, but every day there are more proposals that aim to use the blockchain as a public and

* Corresponding author at: University of the Balearic Islands (UIB), Spain.
E-mail address: xisca.hinarejos@uib.es (M. Francisca Hinarejos).

https://doi.org/10.1016/j.elerap.2019.100869
Received 2 January 2019; Received in revised form 7 June 2019; Accepted 8 June 2019
Available online 14 June 2019

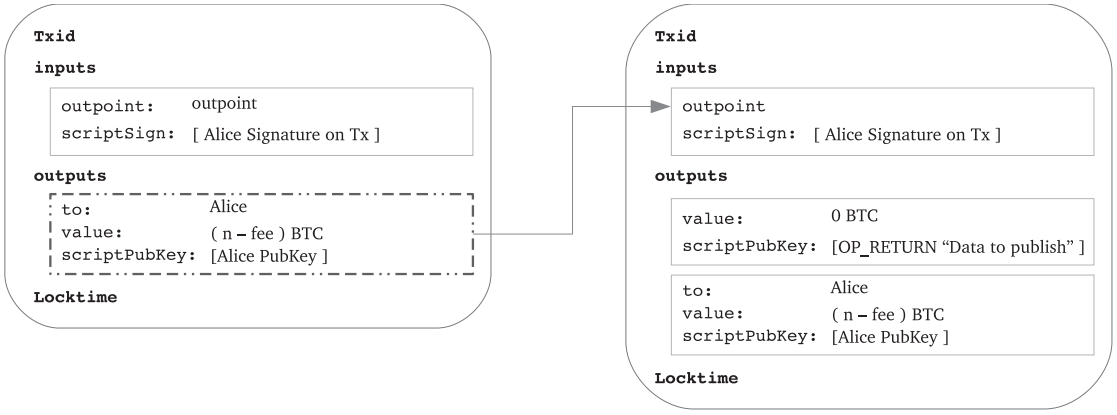

Fig. 1. A high-level picture of a general Bitcoin transaction including an OP_RETURN output.

unalterable record, for multiple and heterogeneous applications (Salah et al., 2019; Zheng et al., 2018): healthcare, energy industry leveraging, energy saving, precision farming, unmanned ocean bed exploration, risk management, internet of things, land registration, education, reputation systems, etc. In relation to contract signing based on blockchain, we only found two proposed protocols, Huang et al. (2017) and Tian et al. (2017), and they are based on smart contracts or scripts. However, our hypothesis goes further, as we propose a protocol that does not use smart contracts or scripts and that allows a blockchain to be used where data can be published in a simple way.

In this paper we present the first solution for contract signing based on a simple use of Bitcoin without using the services of a conventional TTP and without the disadvantages of the classic solutions without a TTP. We could consider that the network of Bitcoin miners act as if they were TTPs. However, unlike the TTPs used thus far, we are no longer facing a single point of failure, and the parties should not agree on a specific TTP. In fact, no single miner is considered trustworthy, but the whole network of miners and the characteristics of Bitcoin provide confidence that the collective as a whole works properly.

2. Publishing data on blockchain

Blockchain provides a solution for digital trust because it allows information to be recorded in a public space and makes it impossible for anyone to remove it. Moreover, the operation of blockchain is conducted in a transparent, time-stamped and decentralized manner. Bitcoin, proposed by Nakamoto (2008), was the origin of the blockchain technology revolution, which is compared to the birth of the Internet. Since the emergence of Bitcoin, many solutions have appeared, some of them based on the structure and operation of Bitcoin (such as Bitcoin Cash, Dash, and so on) and others with a type of operation that is completely different from Bitcoin (such as ZCash, Ethereum, Ripple, and so forth).

The objective of this section is to describe the most relevant aspects, in relation to our proposal, of the blockchain solutions based on Bitcoin and clearly Bitcoin itself.

2.1. Main issues of Bitcoin-based blockchain

Briefly, we can say that Bitcoin is a cryptocurrency that stores blocks of information that are identical across its network. Bitcoin solves the problem of double spending through a distributed P2P ledger, which cannot be controlled by any single entity, and it has no single point of failure.

Each Bitcoin user must have a pair of asymmetric cryptography keys to be able to perform transactions with this cryptocurrency. In fact, each transaction consists of a message in which the owner (A) of coins indicates that she wants to transfer a part of these coins (or all) to another user (B) who has a certain address (or public key); A must sign this message with her private key and send the resulting information to the distributed network, and it will be included in a validated block if it satisfies all the network requirements.

Then, miners, the group of entities that are responsible for validating and publishing the transactions of the users in the blockchain, receive rewards in two main ways: for resolving the proof-of-work established for the current block and the fees paid by each user attempting to publish a transaction (for example, by A). The miners play a key role from the perspective of security. In the case of cryptocurrencies, they guarantee that double spending of a coin cannot occur. In our application, the miners are also a fundamental element for the security of the contract signing protocol. There are two critical aspects that are related to each other that must be addressed: the time between when a user requests the publication of a transaction until it is in fact confirmed and attached to a new block, and the cost of this publication operation. We address both of these aspects later in this paper.

A Bitcoin transaction can generally be divided into two main parts: inputs and outputs (see Fig. 1). Every input indicates an output from a previous transaction from where bitcoins will be spent. Such output will be identified by a transaction id (the hash of the transaction) and a sequential value (since transactions may contain multiple outputs). The outputs of the transaction indicate where the bitcoins from the inputs will be stored as new outputs. Outputs that have not already been spent are known as unspent transaction outputs (UTXO) and they in fact represent the total bitcoins in circulation.

2.2. Data publication

Although the cryptocurrency infrastructure is designed to carry out currency transactions, it also offers the possibility of publishing information that is not very large, acting as a distributed registry. Bitcoin allows for the publication of data using different methods. For example, the P2FKH (*Pay-to-Fake-Key-Hash*) script utilizes the PubKeyHash field of the output script to store 20 bytes of data per output, and several outputs can be included in a single transaction to store more data. Nevertheless, this method is inefficient and can incur a large overhead. Another method is the Coinbase data, which "is the content of the input of a generation transaction" and can be up to 100 bytes in size. However, it is only available to miners (not general Bitcoin users). Finally, the OP_RETURN standard script was included as a response to the increasing number of users using P2FKH to store data (or metadata) in transactions. This method allows a small amount of data (80 bytes) to be included by general Bitcoin users in each transaction. Bitcoin is not the only solution that allows the use of the OP_RETURN field to store or publish small amounts of data. Other cryptocurrencies that work in the same way as Bitcoin (same functionalities, structure, format, and so forth), such as Litecoin and Bitcoin Cash, also allow including data in

the OP_RETURN of a transaction.

In the implementation section, we detail how the OP_RETURN field is used for our contract signing protocol. In addition, because the publication of this information is subject to financial compensation for the entities that are responsible for the publication (the miners), we also analyze the cost of our proposal for some of the Bitcoin-based solutions. We will use historical data because forecasting the future fluctuation of the price of cryptocurrencies is not an easy task, as it can depend on many factors (for example, Mai et al. (2018) analyze the influence of social networks on the value of bitcoins).

3. Related work

Traditionally, solutions for contract signing have been divided into two groups: those without a TTP and those with a TTP. Solutions without a TTP (Goldreich, 1984; Even et al., 1985; Damgård, 1995) 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 proposals were proposed. Solutions without a TTP have commonly been criticized because they involve high computational and/or communications costs.

Solutions with a TTP are subdivided into two large groups: those with an online TTP (Ben-Or et al., 1990) and those with an offline TTP (Asokan et al., 1997; Bao et al., 2004; Ferrer-Gomila et al., 2004). In the solutions with an online TTP, a TTP appears that intervenes in each exchange, and its intervention is necessary to guarantee fairness. This type of solution receives a double criticism: the TTP can become a bottleneck, and it can be difficult for the parties to agree on a TTP that gives them confidence.

To solve the bottleneck problem, authors propose solutions with an offline TTP, also called optimistic protocols. In these solutions, the TTP only intervenes in cases of conflict rather than in each execution. Although these protocols can solve the bottleneck problem if it is assumed that the conflict rate will be low, the problem of agreeing on which TTP the parties must use still remains.

Blockchain, and more specifically, Bitcoin and derived cryptocurrencies, emerged as a different way to solve the security problem of double spending of electronic money. However, as we have already indicated, researchers have proposed alternative uses for blockchain, taking advantage of the fact that its infrastructure provides a public and “unalterable” registry. This would solve the problem of the parties having to agree on a TTP to achieve fairness. In addition, this property has been complemented with the appearance of smart contracts.

(Bentov and Kumaresan, 2014; Jayasinghe et al., 2014; Andrychowicz et al., 2014) are the first references attempting to achieve the fairness requirement using blockchain. Bentov and Kumaresan (2014) and Andrychowicz et al. (2014) introduced the concept of an economic penalty to be executed by a smart contract in the event that a party is dishonest (does not comply with the provisions of the protocol). Bentov and Kumaresan (2014) applied his proposal for fair secure computation and fair lottery. Andrychowicz et al. (2014) made a generic proposal for two-party secure computation, and they stated that one possible application is fair contract signing; however, they did not provide a solution for contract signing. Jayasinghe et al. (2014) presented a protocol for payment for product that guarantees fairness, using Bitcoin as a payment method, and a TTP is involved.

Additionally, we can find other proposed approaches (Liu et al., 2016; Heilman et al., 2016; Delgado-Segura et al., 2017; Zhang et al., 2018; Liu et al., 2018) related to fair payment (with cryptocurrencies) in exchange for data, a receipt or a service. These schemes are based on smart contracts to achieve the fairness requirement (e.g., Liu et al. (2016) on smart contracts for Ethereum, and Delgado-Segura et al. (2017) on the Bitcoin scripting language).

Hasan and Salah (2018) propose a proof of delivery scheme for digital assets, “using Ethereum smart contracts to provide immutable and tamper-proof logs, accountability, and traceability”. The security of their scheme is based on collateral (twice the price of the item to be delivered) to incentivize the actors (customers, provider, file server) to act honestly. The authors claim that their solution eliminates the need of a TTP. However, we can see that one of the actors is an arbitrator that only intervenes in case of dispute (optimistic TTP). The security analysis does not indicate what can happen if this entity acts maliciously. The same authors propose a scheme for proof of delivery of physical assets (Hasan and Salah, 2018), with many similarities to the previous solution: based on Ethereum smart contracts, using collateral to incentivize honesty, and with the existence of an arbitrator to intervene in case of dispute. Very interestingly, the authors provide an estimate of the cost of their proposal in ether (the cryptocurrency of Ethereum).

Therefore, even though these schemes are solutions for payment in exchange for something or for proof of delivery, thereby belonging to the family of fair exchange of value, they are not directly applicable to contract signing.

Finally, we have only found two references (Huang et al., 2017; Tian et al., 2017) that provide protocols for contract signing using blockchain to achieve fairness. Tian et al. (2017) explained a contract signing protocol where contracts are in pdf format, and they are signed by means of a blockchain-based application without TTPs (”the application avoids a single point of failure”). Dishonest parties are monetarily penalized to achieve fairness. The authors emphasized that the use of blockchain does not mean a loss of privacy: ”blockchain nodes do not know the contract content, contract signers’ identities nor contract signatures”.

Huang et al. (2017) presented a fair three-party contract signing protocol based on Bitcoin scripts and on a verifiable encrypted signature. Again, ”a dishonest party is monetarily penalized as it aborts after receiving the current output, and the privacy of the contract content can be preserved on the public chain”. Only the property of fairness was verified, and a performance analysis (referring to the complexity of communication and messages) was provided.

Note that both proposed approaches base their security on two elements: the execution of smart contracts and establishing a penalty for dishonest users. Regarding the use of smart contracts, two possible criticisms can be discussed: it could involve a significant cost for the contract signers, and it may present efficiency problems. Regarding the penalty, it will not always be easy for parties to determine a satisfactory punishment amount, and in any case, users has to have at least that amount of money as a cryptocurrency linked to the used blockchain. Later in this paper, we will compare our proposal with these schemes from the point of view of costs, efficiency, and compliance with security requirements.

In short, we have not found any completely satisfactory solution for the contract signing scenario in the literature.

4. Proposed contract signing protocol

4.1. Overview

Our proposed protocol for fair contract signing is based on blockchain technology and does not require the existence of a conventional TTP. Essentially (see Fig. 2), the two parties attempting to sign a contract must exchange their corresponding non-repudiation evidence regardless of the blockchain network. The contract will be signed when *Alice* has *Bob*’s non-repudiation evidence (*NRR*, non-repudiation of receipt), *Bob* has *Alice*’s non-repudiation evidence (*NRO*, non-repudiation of origin), and there is evidence that both parties are in possession of both the *NRO* and the *NRR*. The objective is for neither party to be in a situation of advantage (or disadvantage) with respect to the other: one party cannot decide unilaterally whether the contract is signed or not. That is, *Alice* (who starts the protocol) sends her non-repudiation evidence (*NRO*) to *Bob*, and *Bob* sends his non-repudiation evidence (*NRR*) to *Alice*. Now, to finish the contract signing in a fair way (reflecting the agreement between *Alice* (*NRO*) and *Bob* (*NRR*)), *Alice* must publish (in

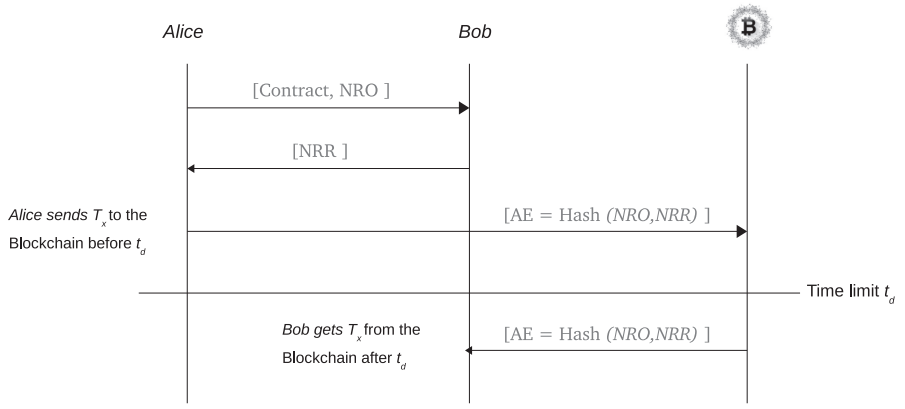

Fig. 2. A sketch of the certified email proposal.

the blockchain network) the acknowledged evidence (*AE*) within a finite time (t_d), and it must be authenticated using the blockchain address set in the contract agreements.

The publication of any transaction (with or without data) in the Bitcoin-based blockchains implies a cost (fee) that must be paid by the entity that makes the transaction. In our protocol, we are assuming that *Alice* starts the protocol and that she has a blockchain address and coins to pay the transaction fees. One wonders whether this situation is desirable. On many occasions, *Alice* may be the most interested party in signing the contract, and thus, she is willing to assume that cost. However, nothing prevents the parties from agreeing on another cost distribution.

4.2. Protocol specifications

The signers, *Alice* (*A*) and *Bob* (*B*), exchange messages and non-repudiation evidence, following the message flow described in Fig. 3.

The notation used in describing the protocol is as follows:

M	contract content
NRO	non-repudiation of origin
NRR	non-repudiation of receipt
AE	acknowledged evidence
t_d	delivery deadline
$h()$	one-way collision-resistant hash function
$Sig_X(x)$	signature on the element x made by party X
Adr_X	blockchain address of the party X
T_x	blockchain transaction

In the first step (Fig. 3 – Step 1), *Alice* sends what can be considered a token of commitment, m_1. It is a document signed by *Alice* which she provides to *Bob* so that he can verify and prove that *Alice* wants to sign the contract. The token consists of the following:

- The content of the contract M to be signed.
- A time value, t_d, that is useful to achieve timeliness, so that neither party has to wait indefinitely to know whether the contract is signed; *Alice* assumes the commitment that she will publish the acknowledged evidence (*AE*) in the blockchain before t_d if she receives the non-repudiation evidence from *Bob* and that if the transaction containing the *AE* is not validated before t_d, then the evidence that may have been sent by *Bob* will be without effect.
- A blockchain address Adr_A from which *A* knows the corresponding private key.
- A signature computed by *Alice* over the previous information (M, t_d and Adr_A) so she will not be able to deny having sent it.

Step 1. $A \rightarrow B$: $m_1 = \{M, t_d, Adr_A, NRO\}$ where $NRO = Sig_A(M, t_d, Adr_A)$

Step 2. $B \rightarrow A$: $m_2 = \{NRR\}$ where $NRR = Sig_B(M, t_d, Adr_A)$

Step 3. $A \rightarrow Blockchain$: $m_3 = \{h(m_1), AE\}$ where $AE = h(NRO, NRR)$

Step 4. $B \Leftrightarrow Blockchain$: $\{h(m_1), AE\}$

Fig. 3. Messages involved during the contract signing between *Alice* and *Bob*.

If *Bob* does not want to sign the contract, he should simply ignore the message received from *Alice*. If he wants to sign it, he must send a message indicating such (Fig. 3 – Step 2) with the following content:

- *Bob* must sign M, t_d and Adr_A, obtaining *NRR*, which is part of the complete non-repudiation evidence.

The *NRR*, sent by *Bob* to *Alice*, confirms that *Bob* wants to sign the contract, as the *NRO* indicates that *Alice* wants to sign the contract *M* with *Bob*. However, the *NRR* and *NRO* are not useful for *Alice* or *Bob* if *Alice* does not publish *AE* as a valid transaction in the blockchain before t_d. Therefore, *Alice* must perform the following actions (Fig. 3 – Step 3):

- Create a blockchain transaction T_x with an `OP_RETURN` output that includes the values $[h(m_1)||AE]$, where $||$ denotes concatenation. The input of the transaction T_x spends bitcoins from address Adr_A indicated in the m_1 data. `OP_RETURN` allows publishing up to 80 bytes. Thus, if we use an SHA-256 hash function, only 64 bytes are required.
- Once T_x is in the blockchain, *Alice* can optionally provide the transaction identifier for T_x to *Bob*. We prefer not to add it to the protocol specification because it does not provide security to the proposed scheme (this message could be lost or may not be sent by *Alice*).

With the publication of T_x in the blockchain before t_d, *Alice* obtains the complete non-repudiation, which is composed of two elements:

- *NRR* received from *Bob*, and
- T_x, which indicates that the transaction is validated and published in a block of the blockchain before t_d and conveying *AE*.

Note that if *Alice* wants to amortize the cost of the transaction, she can use the same transaction to pay other users, but just one `OP_RETURN` per transaction is allowed. Moreover, *Alice* could use the remaining time (from the arrival of *NRR*, t_{nrr}, until the deadline time, t_d) to assess the cost of the transaction. We address this issue later in this paper.

Having the identifier of T_x (or the address Adr_A), *Bob* may retrieve such transaction from the blockchain to obtain *AE* (Fig. 3 – Step 3). For this purpose, *Bob* performs the following steps:

- Retrieve T_x from the blockchain.
- Check that T_x includes an `OP_RETURN` output with the values $[h(m_1)\|AE]$.
- Verify that T_x has been correctly signed by *Alice*. Strictly speaking, such validation is not necessary because the fact that T_x is included in the blockchain implies that the verification has already been performed by the blockchain network, that is, the input of the transaction T_x corresponds to the address Adr_A.
- Verify that $h(m_1)$ computed from m_1 received in *Step 1* is equal to $h(m_1)$ extracted from T_x.
- Retrieve *AE* and validate that $h(NRO, NRR)$ computed from *NRO* and *NRR* (*Steps 1 and 2*) is equal to *AE* extracted from T_x.

After retrieving T_x from the blockchain, *Bob* obtains the complete non-repudiation evidence, which is composed of two elements:

- *NRO* received from *Alice*, and
- T_x, which indicates that the transaction is validated and published in a block of the blockchain before t_d and conveying *AE*.

5. Security analysis

In this section, we prove that our proposed protocol satisfies the following security requirements (Asokan et al., 1998): fairness, timeliness, confidentiality, and non-repudiation (of origin and of receipt).

5.1. Fairness

A contract signing protocol must fulfill the fairness requirement. After execution of the protocol (complete or partial), both *Alice* and *Bob* should be able to obtain evidence that the contract is signed, or neither of them should be able to do so.

Therefore, we must prove that our protocol is fair for an honest party, irrespective of the actions performed by a dishonest party or even an external attacker. *Alice* will be able to prove that the contract is signed if she possesses *NRR* (received from *Bob*), and *AE* is published in the blockchain before t_d. *Bob* will be able to prove that the contract is signed if he possesses *NRO* (received from *Alice*), and *AE* is published in the blockchain before t_d. Now, we will analyze the possible attacks after each step of the protocol.

After *Alice* sends m_1 (which contains *NRO*) to *Bob*, no attack can be performed. *Bob* cannot obtain the complete non-repudiation evidence because he needs *AE* (and only an *AE* published by *Alice* is valid), and *Alice* does not have the non-repudiation evidence from *Bob*. If *Bob* does not send *NRR* to *Alice*, then the execution of the protocol ends, and neither of them will have evidence that the contract is signed (because, in fact, the contract has not been signed). Therefore, the situation is fair for both *Alice* and *Bob*.

After *Bob* sends *NRR* to *Alice*, no attack can be performed. If *Bob* sends an incorrect *NRR* then *Alice* must stop the exchange, and none of them will have evidence that the contract is signed (again, in fact, it is not signed). Even if it is correct, *Bob* cannot obtain the complete non-repudiation evidence because he does not have *AE*. Meanwhile, *Alice* does not have the complete receipt: *AE* must be validated and published in the blockchain. If *Alice* does not send *AE* (next step of the protocol) then the execution of the protocol ends without the contract being signed. *Alice* and *Bob* have no evidence to prove otherwise. Therefore, this situation is fair for *Alice* and *Bob*.

After *Alice* sends a valid *AE* to the blockchain that is validated and published before t_d, no attack can be performed. After this step, *Alice* has the complete non-repudiation of receipt (*NRR* and the publishing in the blockchain), and *Bob* can download *AE* from the blockchain and obtain the complete non-repudiation evidence *NRO* plus *AE*. Therefore, this situation is fair for *Alice* and *Bob*.

If *Alice* sends an invalid *AE*, *Bob* will not be able to obtain the valid and complete non-repudiation evidence, but he will be able to prove it. The value of *AE* is signed by *Alice* and published in the blockchain. If *AE* is incorrect, *Bob* can prove it, and a judge can determine that the contract *M* has not been signed.

If *AE* is published after t_d, the contract will be considered not signed, and the situation is again fair for *Alice* and *Bob*: neither of them has complete non-repudiation evidence. Therefore, the protocol remains fair even if the blockchain infrastructure (a group of dishonest miners) delays the publishing of *AE* after t_d, or even if a miner makes a sybil or an eclipse attack and does not allow *AE* to be published. In a sybil attack (Douceur, 2002), an attacker deploys a group of nodes trying to control a part of the Bitcoin network. Bitcoin is able to overcome a sybil attack through the consensus algorithms, which protect the network as a whole. Nevertheless, it does not prevent individual nodes from being isolated from the network under an eclipse attack (Heilman et al., 2015), for example, preventing *Alice* from publishing *AE*. However, there are also measures to prevent this type of attack, for example, connecting *Alice* to a set of known and trusted peers, among other solutions (Conti et al., 2018).

Therefore, the fact that *AE* is published after the deadline t_d or not published, either because of *Alice* or because of the miners, does not cause an unfair situation for *Alice* or *Bob*.

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

5.2. Timeliness

A contract signing protocol must fulfill the timeliness requirement: “honest entities can unilaterally choose to terminate the protocol in a finite amount of time without losing fairness”. In our protocol, 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 *AE* is published in the blockchain), but the parties always know the final state of the exchange no later than t_d. Therefore, the timeliness requirement is satisfied.

5.3. Confidentiality

A contract signing protocol should optionally (users’ decision) meet the confidentiality requirement. If *Alice* and/or *Bob* want this requirement to be satisfied, then ”for the correct and fair protocol execution, no third entities need access to the contract content”. *Alice* can send the content of the contract *M* encrypted with a symmetric key *k*, and this key *k* is encrypted with the public key of *Bob*, PU_B. In this way, the content of the contract would only be known by *Alice* and *Bob*, who are the only ones capable of obtaining the symmetric key *k*. We have not reflected this situation in the protocol such that it remains as simple as possible.

Thus, we can confirm that the protocol can easily satisfy the confidentiality requirement.

5.4. Non-repudiation

A contract signing protocol must provide NRO service for the recipient and NRR for the originator. ”Evidence of origin/receipt must be provided that will allow the recipient/originator to demonstrate to an arbiter whether the originator/recipient signed the contract”. During our protocol execution, evidence is generated as non-repudiation proofs for *Alice* and *Bob*. In particular, the protocol offers the following:

- Complete non-repudiation of origin. *Alice* provides two signatures to *Bob*. In the first step of the protocol, she provides *NRO*. Therefore, she cannot deny having sent the contract *M* and the corresponding

signature *NRO*. In the third step, *Alice* signs the value *AE* as part of a blockchain transaction. Therefore, she cannot deny having sent that value *AE* to the blockchain. Consequently, she cannot deny having sent that information (*M*, *NRO* and *AE*), thus satisfying the non-repudiation of origin requirement.

- Complete non-repudiation of receipt. *Bob* provides one signature to *Alice*. In the second step, he signs (*NRR*) the contract *M*. Therefore, he cannot deny having received it and that he wanted the contract to be signed. The protocol specification indicates that the publication of the value *AE* 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 *AE* is published before t_d, then *Bob* cannot deny having signed the contract, thus satisfying the NRR requirement.

6. Assessing the cost of the solution using a Bitcoin-based blockchain

Bitcoin and similar cryptocurrencies incentive miners using two mechanisms: block confirmation (reward for solving the next block) and fees (attached to any transaction on that block). Because the size of a single block is limited, transactions have to wait in a pool of unconfirmed transactions until they will be included in a block. Moreover, because it is known that miners prioritize transactions with higher fees, the time for including a transaction in a block may depend on the amount of fees applied to a transaction. Therefore, it is important to analyze the cost to include a transaction in a block in a reasonable period of time with the minimum amount of fees. In our particular case, it means that we need to estimate the fees to be included in a transaction such that *AE* can be published before t_d.

The transaction fee depends on two factors: the transaction length (size) and the fee density (also known as fee rate per published byte). For our purpose, the size of the transaction can be considered fixed (as we explain later), whereas the fee density is variable and depends on the number of transactions waiting to be confirmed.

For our study, we consider five blockchain networks based on the same Bitcoin operation and structure (which are spin-offs or hard-forks from Bitcoin), supporting the `OP_RETURN` output and ranked in the top 20 cryptocurrencies[1]: Bitcoin, Litecoin, Bitcoin Cash, Bitcoin Gold and Dash. Table 1 summarizes the main parameters of these blockchain networks with an influence over our study.

6.1. Transaction size

All Bitcoin-based networks share the same transaction format and structure, occasionally called *legacy transaction*. However, some of them include another transaction format known as *segwit transaction*. The segregated witness mechanism (supporting the use of *segwit transactions*) offers some improvements over the use of *legacy transactions*, such as reductions in both the transaction length (resulting in a lower fee) and the time of transaction validation (resulting in faster transactions) (see Bitcoin-Magazine (2017) for more information). Since the fee to be paid depends on the size of the transaction, we next analyze the size of a transaction expressed in both transaction formats (*legacy* and *segwit*) considering the requirements of our proposed protocol and the support of both formats for the considered blockchain networks (see Table 1).

For both transaction formats (see Table 2 and Table 3), the size of a transaction can be divided into a variable length (depending on the number of inputs and outputs included) and a fixed length called overhead (version, length and code fields). First, recall that the `OP_RETURN` data must include the data sent by *Alice* to the blockchain, that is, $[h(m_1)||AE]$, where both $h(m_1)$ and *AE* are hash values. Considering an SHA-256 hash function that provides a suitable security level with hash output length of 32 bytes, $[h(m_1)||AE]$ requires 64 bytes (see Table 2 and Table 3). As shown in Table 1, the maximum length that an `OP_RETURN` can convey as data in the considered blockchains is larger than the size that we require to publish in our proposed protocol.

A *legacy transaction* (see Table 2) to publish our evidence requires a minimum size of 267 bytes: one P2PKH[2] input script (normally requires 107 bytes), a P2PKH output script (25 bytes), an `OP_RETURN` data (64 bytes), and all the bytes of overhead of the code and length fields. If we evaluate the size when a *segwit transaction* is used (see Table 3), then we obtain the same transaction size calculated for the *legacy transaction*, but in this case, only the virtual size has to be considered to assess the fee (BitcoinCore, 2019). As specified in BitcoinCore (2019), the virtual size can be calculated as follows:

$$vsize = \lceil (Btx^* \ 3 + Ttx)/4 \rceil$$

where *Btx* is the base transaction size (without marker, flag and witness fields) and *Ttx* is the complete transaction size (including all fields - same size as a *legacy transaction*). With these data, the virtual size of the transaction is 186 bytes; therefore, the use of a *segwit transaction* reduces the required transaction size by approximately 30% in our proposed protocol. However, this mechanism is not supported by Bitcoin Cash and Dash (see Table 1).

6.2. Transaction fee

As explained at the beginning of this section, the applied fee depends on the size of the transaction (bytes) and on the fee density. Thus, once we have calculated the required transaction size to publish the non-repudiation acknowledgment with both *legacy* and *segwit transactions*, we must assess the cost to be paid for this transaction by analyzing current and past expenditures of publishing transactions.

Table 4 shows the current average fee densities[3] (in US dollars) required to publish in each blockchain and the corresponding cost to publish data using a *legacy transaction* in our protocol. As shown, this is a reasonable price to use the publication service.

To assess the cost viability of the solution in the past, we consider a period of two years, in which these cryptocurrencies have fluctuated enough upward and downward. For this purpose, we use the dataset provided by Coin-Metrics (2019), which includes statistical data for all the blockchain networks under analysis. The dataset includes the average values of several parameters for each day, but it does not include the fee density. However, it provides the total fee paid by all-day transactions and the total size in bytes of the published blocks; thus, we can estimate the average fee rate per day. Because the fees are specified in the particular cryptocurrency (bitcoins, litecoins, and so forth) and the dataset also specifies the exchange rate among them and US dollars (per day), we evaluate the fee rate in US dollars to conduct a unified comparison of the publication cost for the non-repudiation acknowledgment using each of the considered blockchain networks.

Fig. 4 depicts the average fee that our protocol would have paid per day during the past two years if *legacy transaction* is used. Note that the y-axis on the right depicts the price using Bitcoin, and the y-axis on the left presents the price for the remaining blockchains. Analyzing Fig. 4, we detect a clear peak in fees, which is basically due to congestion events and because of an impressive increase in the trading price of Bitcoin.[4] Taking this unfavorable scenario into account, the maximum cost of publication in our protocol considering the Bitcoin network would have been quite high (approximately $27), but at the same time,

[1] Top 100 Coins by Market Capitalizationhttps://coinmarketcap.com/coins/ accessed: 2018-12-10

[2] "Pay-to-PubKey-Hash (P2PKH) is the basic form of making a transaction and is the most common form of transaction on the Bitcoin network". Source:https://en.bitcoinwiki.org/wiki/Pay-to-Pubkey_Hash

[3] Data publication date: 2018/11/21.

[4] https://markets.businessinsider.com/currencies/btc-usd.

Table 1
Statistics of the main parameters of the Bitcoin-based blockchain solutions.

	Bitcoin	Litecoin	Bitcoin Cash	Bitcoin Gold	Dash
OP_RETURN data size	80	80	220	80	80
Avg. fee/byte (last two years)	0.00537804	0.00018029	0.0001712	0.00002373	0.00017628
Avg. Block Confirmation	10 min	2.5 min	10 min	10 min	10 min
Segwit support	yes	yes	no	yes	no

Table 2
Legacy transaction specified for our proposed protocol.

Field	Size (bytes)
version	4
input count	1
outpoint	36
P2PKH script length	1
P2PKH input script	107
sequence	4
output count	1
import value	8
P2PKH script length	1
P2PKH output script	25
import value	8
OP_RETURN script length	1
OP_RETURN code	1
push data	1
OP_RETURN data ($h(m_1)\|\|AE$)	64
lockTime	4

Table 3
Segwit transaction specified for our proposed protocol.

Field	Size (bytes)
version	4
marker	1
flag	1
input count	1
outpoint	36
script length	1
sequence	4
output count	1
import value	8
P2WPKH script length	1
P2WPKH output script	22
import value	8
OP_RETURN script length	1
OP_RETURN code	1
push data	1
OP_RETURN data ($h(m_1)\|\|AE$)	64
version	1
witness length	1
witness signature	71
witness length	1
witness public key	33
lockTime	4

such cost would have been kept quite low in the remaining blockchains (see Table 5). For example, transaction fees were under $0.65 in Litecoin, and Bitcoin Gold was the cheapest one with a transaction fee below $0.035 during the peak. Regardless, despite the peak rates, the average fees per transaction are very reasonable.

The support for *segwit transactions* was adopted by some blockchain networks at different times in 2017,[5] and its use has been progressively increasing. In fact, since August 2018, the use of this type of transaction is over 40% of all payments.[6] To analyze the effect of using *segwit transactions*, we consider a period of one year (see Fig. 5) where the segwit mechanism was completely activated in all considered blockchains. As expected, the maximum cost is reduced by 30% (Table 5) due to the reduction in the transaction size.

Now, assuming that *Alice* is a regular user of cryptocurrencies (i.e., she creates transactions on a regular basis), we evaluate the cost of adding an OP_RETURN payload to a regular transaction. In this case, she can leverage one of her regular transactions to publish data, and only the OP_RETURN payload (its size) is assumed to be an additional cost. Fig. 6 shows the cost due only to the OP_RETURN additional data. Table 5 summarizes the estimated maximum cost for publishing data in the different situations (*legacy transaction*, *segwit transaction* and OP_RETURN payload).

Finally, the transaction fee can also be further reduced if its publication can be delayed. Recall that a block confirmation measured in minutes is not the same for each blockchain (see Table 1). For example, in Litecoin, it is 2.5 min on average, and it is 10 min for the remaining solutions. Considering current statistics[7] about the fee/byte paid if a transaction is delayed a few blocks, a Bitcoin transaction cost can be reduced tenfold (see Table 6).

7. Proof of concept implementation

In this section, we provide a proof of concept (POC) implementation that demonstrates the viability of implementing and deploying the proposed protocol in a real scenario with the support of a working blockchain network. Among the analyzed blockchain networks with OP_RETURN opcode support, we use the Bitcoin network to implement the POC because it is one of the most widespread blockchains, with an excellent availability of tools and libraries to support the rapid prototype development of services and applications on top of a blockchain network. To avoid spending actual money, we ran and tested the implementation using Bitcoin's test network, called Testnet3 at the time of writing. Indeed, developing on top of the Testnet3 network means that all code is going to work in the same way when it is deployed to the main production network because the Testnet3 network uses exactly the same operating parameters, transactions and so on as Mainnet, with the principal differences being that the coins in circulation have no trading value and the network difficulty to resolve new blocks is much lower than in Mainnet.

The POC has been developed using the Java language to implement all the protocol behavior and code to publish data to the blockchain. We have relied on the use of the following third-party libraries and tools to implement and verify the POC:

- Bitcoinj library (Bitcoinj, 2019). This library exposes a complete API to interact without intermediaries with Bitcoin (both Mainnet and Testnet3): generate, sign, verify and publish transactions; manage funds with an integrated wallet; and so forth.
- BlockCypher Bitcoin online explorer and API (Blockchyper, 2019). This

[5] Activated in April for Litecoin and in August for Bitcoin and Bitcoin Gold.
[6] https://transactionfee.info/charts/payments/segwit.
[7] Data publication date: 2018/11/26. Source:https://live.blockcypher.com/.

Table 4
Current fees for a *legacy transaction* in each blockchain network.

	Bitcoin	Litecoin	Bitcoin Cash	Bitcoin Gold	Dash
Avg. fee density (price/byte)	$0.0017505126	$0.00003758	$0.00001078	$0.00000396	$0.00003841
Avg. Tx fee in our protocol	$0.4673	$0.01	$0.0028	$0.001	$0.01025

online service allows developers to explore the Testnet3 blockchain to search for data stored in it in the same way as Mainnet explorers do. It provides a powerful API to look up data in the blockchain.

For the implementation, we assume that it is unfeasible for any party involved in the exchange to download and keep synchronized the full blockchain data since parties may be using mobile devices with a

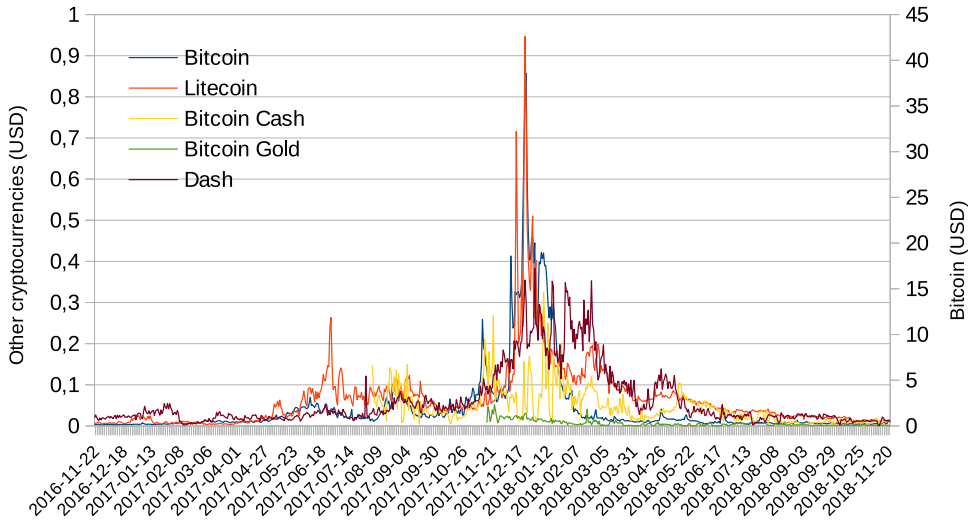

Fig. 4. Estimated average cost of our protocol (in USD) over the past two years using the *legacy transaction*. A comparison among different blockchains.

Table 5
Maximum estimated cost of our protocol in the past considering the *legacy transaction*, the *segwit transaction* and only the `OP_RETURN` for each considered blockchain network.

	Bitcoin	Litecoin	Bitcoin Cash	Bitcoin Gold	Dash
Maximum *legacy transaction* cost	$38.56	$0.946	$0.324	$0.051	$0.383
Maximum *segwit transaction* cost	$26.86	$0.659	–	$0.035	–
Maximum `OP_RETURN` payload cost	$10.83	$0.265	$0.091	$0.014	$0.107

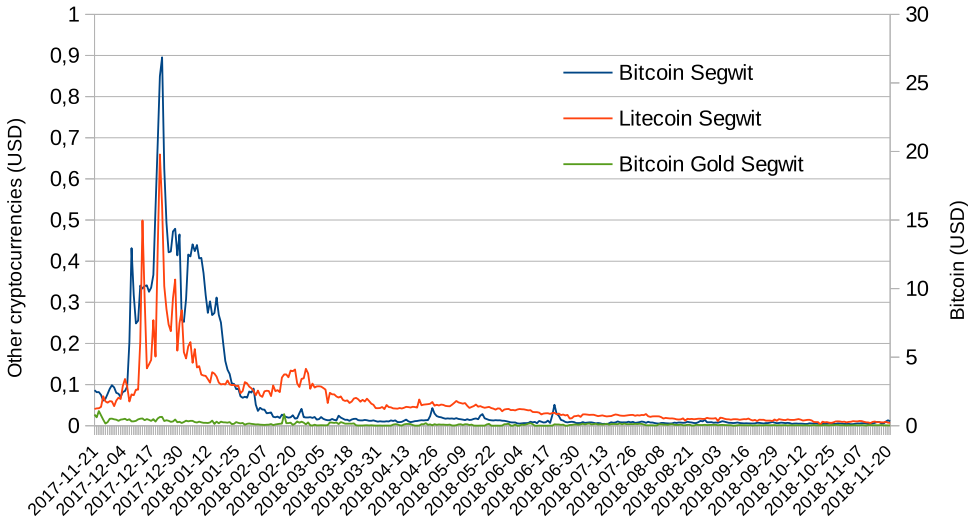

Fig. 5. Estimated average cost of our protocol (in USD) over the past year using the *segwit transaction*. A comparison among different blockchains.

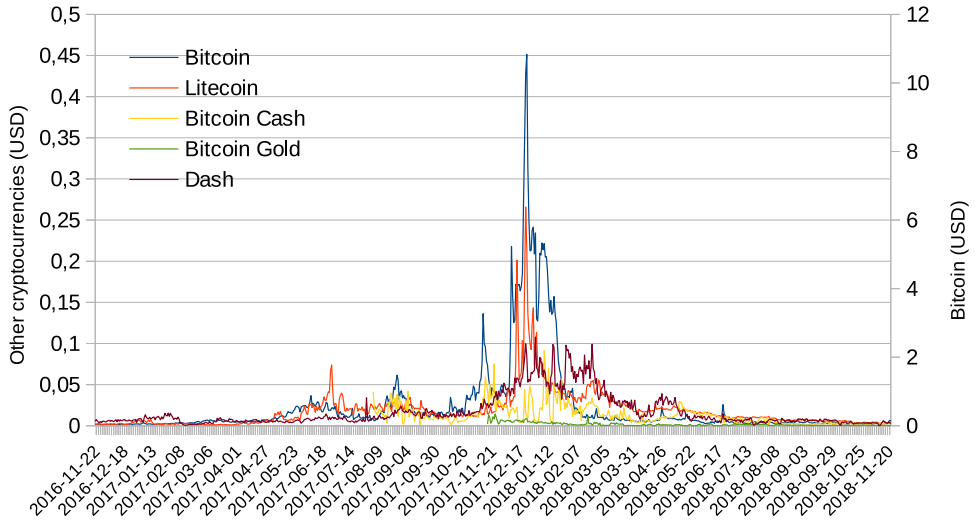

Fig. 6. The estimated average `OP_RETURN` cost in our proposed protocol (in USD) over the past two years. A comparison among different blockchains.

Table 6
Statistics of the fee for the considered blockchains.

	Bitcoin	Litecoin	Bitcoin Cash	Bitcoin Gold	Dash
Fee (1–2 block)	0.34369957	0.007267219	NA	NA	0.0008908
Fee (3–6 block)	0.07811354	0.00058137	NA	NA	0.00089085
Fee (7 + block)	0.02343406	0.00017441	NA	NA	0.00053451

NA – Data not available

limited amount of storage space and bandwidth. *Alice* is specifically sensible to this since she has to build and send a transaction to the blockchain network; thus, she has a write role facing the blockchain. Therefore, we have identified two possible implementation solutions to overcome this limitation:

- Commit a transaction after signing it locally to a third-party service who will be in charge of flooding this transaction to network nodes.
- Implement support for simple payment verification (SPV), meaning that a Bitcoin client can interact with blockchain by only downloading a small part of the longest chain, resulting in a huge storage savings.

Although there are third-party services providing the former solution (Blockchyper, 2019) or it can even be implemented by a server, we opted in favor of the SPV solution, through the use of the Bitcoinj library, to keep the POC simple and independent of any third party to commit transactions to the blockchain network.

In contrast to *Alice*, *Bob* only has a read-only role in relation to the blockchain because he only has to search for the transaction published by *Alice*. Thus, in this case, we use the BlockCypher Bitcoin explorer API service to gather the transaction published by *Alice*.

7.1. Protocol execution example proof using Bitcoin Testnet3 network

In this section, we explain the steps that *Alice* follows to publish the transaction in the blockchain and how *Bob* can search the required transaction and access the data published.

First, *Alice* builds the transaction from the following data:

- Contract to sign (M). Let us suppose that the contract to be signed is the Bitcoin original paper (Nakamoto, 2008). This POC hashes the contract to be signed using the SHA-256 secure collision-resistant hashing algorithm since it is the most secure SHA-2 hashing algorithm that satisfies both security and `OP_RETURN` capacity constraints to store two hashes.
- *Alice*'s Bitcoin address (Adr_A). A random Bitcoin address selected by *Alice*, in which she must have some funds to be able to publish a transaction into the blockchain. To fill this address with some funds, before executing any protocol step, we transferred to this address some free test bitcoins from a public faucet (Bitcoin Testnet3 free faucet, 2019).
- Deadline (t_d). An arbitrary timestamp setting the execution deadline.

With these input parameters, the developed application creates the signatures (non-repudiation evidence), includes $[h(m_1)||AE]$ in the `OP_RETURN` data field, and sends the transaction to the blockchain. Table 7 summarizes the practical values used for each of the above input parameters, together with the result obtained after a complete protocol execution: the published transaction identifier and the hexadecimal encoded string of the data published as `OP_RETURN`. Fig. 7 shows a screenshot from the BlockCypher explorer service website proving that the transaction built by *Alice* has been accepted and successfully published on Testnet3.[8]

Note that the protocol specification indicates that *Alice* has to publish the data before t_d to avoid *Bob* denying he has signed the contract. Thus, *Bob* has to verify whether the transaction was published by *Alice* in the blockchain before t_d and whether it is correct. Recall that the transaction (T_x) (and thus its content) is part of the non-repudiation evidence. In this POC implementation, *Bob* obtains the data from the transaction published by *Alice* using the BlockCypher service API (Blockchyper, 2019). We have two options to retrieve data published on behalf of *Bob* depending on whether *Alice* communicates to *Bob* the

[8] https://live.blockcypher.com/btc-testnet/tx/219e731f4d110a7d87542516a060550f7d774a2a85793df6bbf0a4062696222b/.

Table 7
Execution example: input parameters and data results.

	Parameter	Value
INPUT	M	`b892aad83cd0360c55998cb710b3ff32a00321ca6a5fd61b602093c403c3d161`
	Adr_A	`mirpNr9Zpf58A91eYZc3DnamWPZXgZW8v`
	t_d	1546160400000 (December 30, 2018–09:00:00 AM GMT)
RESULT	T_x	`219e731f4d110a7d87542516a060550f7d774a2a85793df6bbf0a4062696222b`
	`OP_RETURN` payload	`8c8f5c161798085bcda37c10ea861c562281718 9d16e141b5e8e8b51ee103713b694022a0aa22aed58bdaaf9b9b01717535fb4b720abaf592c78bbb56e81e6c`

transaction identifier (T_x):

- *Bob* knows T_x. This is the best case since *Bob* can send a simple API call to the blockchain explorer service with this identifier to receive the complete transaction details, one of them being the content of the `OP_RETURN` output. Upon receipt of the API response, *Bob* can verify its content, checking evidence correctness by applying the procedure described in the protocol specifications.
- *Bob* does not know T_x. In this case, *Bob* has to ask the blockchain explorer for the list of transactions from *Alice*'s address (Adr_A), taking into account that multiple transactions coming from the same address may exist. Thus, code must iterate over the list, checking which one has an `OP_RETURN` transaction output whose content is equal to the expected value based on the data exchanged during protocol execution (recall that *Bob* can compute `OP_RETURN` content data himself since it is composed of data received from and submitted to *Alice*). Once the transaction data meet these requirements, *Bob* can save the T_x identifier as a part of the non-repudiation evidence.

To summarize, we have performed a POC of the protocol execution with practical values to publish data to the blockchain that can be verified by anyone at any time. To do so, we also provide a GitLab public repository[9] that contains scripts and instructions to allow readers to easily verify the correctness of the evidence published on Testnet3 based on the selected input parameters.

8. Ethereum: an alternative to Bitcoin-based solutions

Ethereum is the second largest cryptocurrency by market share according to CoinMarketCap.[10] Similar to Bitcoin, in Ethereum, the nodes validate blocks (composed of transactions) in exchange for rewards; in this case, the cryptocurrency unit is the *ether*.

However, Ethereum was designed as a platform to deploy smart contracts. A smart contract is an executable code that runs on the blockchain to facilitate, execute and enforce the terms of an agreement between parties (Wood, 2017). Thus, a smart contract defined in Ethereum should not be confused with a contract that outlines the terms of a relationship (usually one enforceable by law).

Remember that we are not considering the enforcement of the signed contract between the parties. Although an Ethereum smart contract can be used for our solution, this is not the best option (as we explain later). However, Ethereum allows data to be included in an optional field known as `Data`, which could be considered the equivalent of the `OP_RETURN` field used in Bitcoin. Thus, Ethereum allows data to be published as Bitcoin-based solution does, and our proposal can use this functionality to publish the AE in the same way that we use Bitcoin.

8.1. Assessing the cost of the solution using Ethereum

Similar to bitcoin, the ether is traded on public exchanges, and its price fluctuates daily. If ether is used to pay fees, then the cost of a transaction could be high on certain days and low on others. *Gas*, the internal currency of Ethereum, helps to alleviate this problem because for a transaction, any execution and resource utilization cost is predetermined in Ethereum in terms of gas units (also known as gas cost). Thus, gas is the unit for measuring the computational work of running transactions or smart contracts in the Ethereum network. For example, a general transaction (money transfer) requires a fixed 21, 000 units of gas to be processed.

So, if the cost of a transaction appears to be fixed, how do the miners

[9] https://gitlab.com/secom-uib/contract-signing-btc.

[10] CoinMarketCap: Cryptocurrency Market Capitalizations. Data publication date: 2019/05/01. Source:https://coinmarketcap.com/.

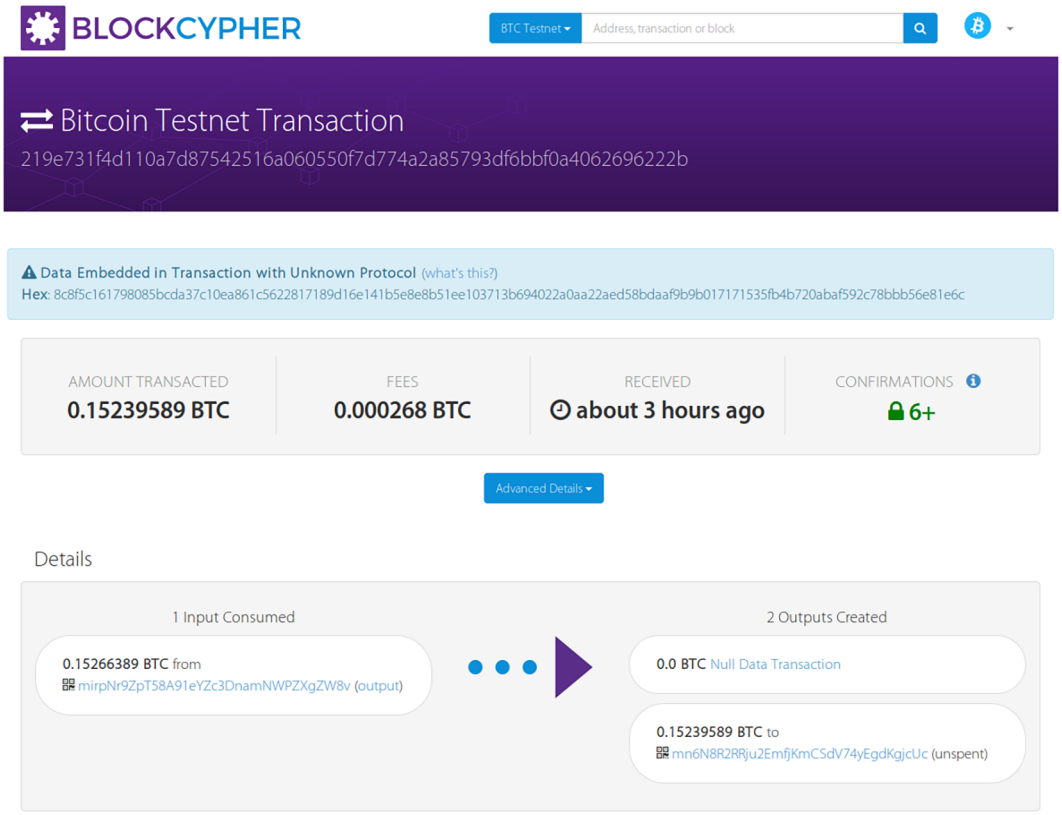

Fig. 7. Proof of example Bitcoin Testnet3 transaction captured from BlockCypher blockchain explorer.

decide which transaction to process first? In fact, the total cost of a transaction in Ethereum (transaction fee), depends on two factors: the gas limit and the gas price. The gas limit refers to the maximum amount of gas a user is willing to spend on a particular transaction, and the gas price refers to the amount of ether a user is willing to pay for every unit of gas. Therefore, the gas price allows users to adjust the transaction price, lowering the price when the ether price increases and increasing the price when the ether price decreases.

8.1.1. Gas limit

As explained above, the standard gas limit for a regular transactions is fixed, in this case to 21, 000 units. However, this limit must be increased if a transaction includes the optional `Data` field. According to the Ethereum yellow paper (Wood, 2017), the gas limit must be increased by 68 gas units for each byte of data included in the `Data` field.

To publish *AE* in our proposal, two mechanisms or options may be used. First, the sender can re-use a general money transaction to include the evidence or generate a new transaction transfer to convey the evidence, which are both similar to the Bitcoin-based solution already explained. Alternatively, a very simple smart contract can be launched by the sender to publish the evidence.

In the first case, *Alice* only needs a general transaction including the 64 bytes of the evidence ($gasLimit = 25,352$). In the second case, *Alice* must conduct two transactions, one to launch the smart contract and a second one to publish the evidence (using a call to the corresponding function of the smart contract). Taking into account that the minimum gas paid for a given transaction is 21,000 and for contract creation is 32,000 (without calling any function of the contract), it seems clear that the best solution (considering cost) is to use a simple transaction to publish the evidence.

8.2. Publication cost: Ethereum versus Bitcoin-based solutions

To assess the cost viability of using Ethereum, we consider the same period of time analyzed for the Bitcoin-based solutions. For this purpose, we use the dataset provided by Etherscan (2019), which includes the average value of the gas price paid each day, measured in Wei, the smallest denomination of ether (1 ether $= 10^8$ Wei). Using the exchange rate between Ether and US dollars (per day), we evaluate the publication cost for *AE*.

Table 8 shows the statistics for the estimated cost of our solution when an Ethereum transaction is used versus the use of *segwit* Bitcoin-based transactions. As shown in the table, Ethereum has a lower cost compared to Bitcoin but has a higher cost compared to the other Bitcoin-based solutions.

9. Comparison with previous solutions

In this section we will compare our proposal with the only two solutions published (Huang et al., 2017; Tian et al., 2017) to date for contract signing based on blockchain. This comparison will focus on the aspects of costs, efficiency, and compliance with security requirements.

We will start with the scheme proposed by Huang et al. (2017). The first aspect analyzed is security, and the authors affirm that their scheme fulfills the properties of fairness, timeliness, non-repudiation and confidentiality, that is, the same properties fulfilled by our proposal. However, if we analyze in detail their proposal, we observe that

Table 8
Statistics for the estimated cost of our protocol in the past. A comparison between *segwit* Bitcoin-based solutions and Ethereum.

		Ethereum	Bitcoin	Litecoin	Bitcoin Gold
Transaction cost	Maximum	$3.0221	$26.864	$0.6594	$0.0358
	Average	$0.1286	$0.9739	$0.0335	$0.0044
Data cost	Maximum	$0.518	$10.83	$0.265	$0.014
	Average	$0.0269	$0.3927	$0.0135	$0.0017

fairness and timeliness properties are not satisfactorily fulfilled. The authors do not address the temporal issue of publication in the blockchain, and more specifically, they do not analyze what happens if there is a delay in the publication even if the signers have acted diligently (period of congestion, high price of publication, etc.). If one of the signatories sends the required data to be published in the blockchain, and this publication suffers a delay greater than an established deadline, in their scheme, the signatory will be "accused" of dishonesty and will be economically penalized.

Our protocol is more robust in this regard. If there is a delay in the publication of the transaction in the blockchain, either because *Alice* (the first signer) is dishonest or due to the actions of the miners, neither party is penalized, and neither party has proof of the contract. In this sense, Huan et al. should extend the explanation to clarify what happens when a signer X already has the signature of signer Y, and X does not deliver his/her signature to Y. In the specification, it is clear that a "dishonest" signer will be penalized economically, but can he/she use the proof that the other party has signed the contract? Does the honest party have proof that the signing of the contract was canceled? These questions should be answered more clearly because otherwise, it cannot be said that the proposal fulfills the fairness and timeliness requirements.

Related to the time required for the publication of transactions in the blockchain, we have the issue of the publication cost. The authors omit the cost of their solution, but we can affirm that their proposal is more expensive than ours. To make the comparison with this proposal, we had to adapt it because it is a scheme for contract signing between three parties, and ours is only for two parties. In their scheme, the two signatories must publish a transaction in the blockchain that contains a secret, while in ours, only one of the signers (*Alice*) must publish a value in the blockchain.

Regarding efficiency, our scheme requires two signed messages be sent (one by each party) and one transaction published to the blockchain. In Huan et al.'s proposal, adapted for two parties, each party sends two messages and publishes two transactions to the blockchain (a deposit and a publication of a secret claiming the deposit previously made). Therefore, Huan et al.'s proposal requires a total of four messages and four transactions.

The comparison with the proposal of Tian et al. is still more complicated because these authors do not use a public blockchain but instead a new variant of Bitcoin (with new script opcodes, new blocksize, etc.). Therefore, we do not know the cost of putting this blockchain in operation or the cost of publishing a transaction on it.

As far as security is concerned, Tian et al.'s solution suffers from problems similar to Huan et al.'s regarding fairness and timeliness. The authors only consider the possibility of a delay due to dishonest behavior by the signatories but not one due to publication in the blockchain. Moreover, they also omit what happens if one of the parties obtains the signature of the other party and does not provide his/her "signed copy". We know that the dishonest party will pay a penalty, but is the signing of the contract invalidated?.

Their proposal fulfills an unusual property in the articles of contract signing: the privacy of the identity of the signatories. To achieve this property, each signer must generate a self-public key certificate that allows "hiding" their identity in the operations with the blockchain. Similarly, our proposal does not allow an attacker to obtain the identity of the two signatories, since only one of the parties publishes (and therefore signs) a transaction in the blockchain. Therefore, an attacker at most may know that an entity (*Alice*) is signing a contract, but it will not know the identity of the other party because with the information published in the blockchain, the identity of the other party (*Bob*) cannot be disclosed.

Regarding the other properties (timeliness, confidentiality, and non-repudiation), the authors do not even mention them much less prove their fulfillment. As in the case of Huan et al.'s proposal, the authors should clarify the indicated problem regarding fairness because it could have consequences for timeliness and non-repudiation. In addition, Tian et al.'s proposal has the disadvantage that a TTP (the minter) is necessary.

Working with a blockchain that is not public (and therefore, for which we lack data) makes it difficult to conduct a cost analysis. But in any case, assuming we implement our solution in their blockchain, we can say that their scheme is more expensive than ours. In their scheme, three transactions must be published, while in ours, only one is needed. In terms of efficiency, the authors provide a performance analysis based on two parameters: the message transmission time (t_{comm}) and the time to perform a modular exponentiation (t_{me}). Their scheme requires five t_{comm} and twenty t_{me}, while our proposal requires three t_{comm} and three t_{me}, that is, our solution is more efficient.

In conclusion, we can affirm that our proposal improves the two previous solutions in terms of costs, efficiency, and compliance with security requirements.

10. Discussion

10.1. Findings and contributions

We highlight five main contributions of this work. Firstly, we have shown that from a theoretical point of view, we can use the blockchain in a simple way to achieve compliance with the essential fairness requirement of the protocols for contract signing (in addition to the other properties of interest). In this way, it is not necessary for the signatory parties to agree on a TTP. We also want to emphasize the simplicity of the use we make of the blockchain and its contrast with the protocols based on smart contracts, which are a very powerful tool for solving complex security problems, but with a possible drawback: the economic cost implied by the execution of the smart contract.

Secondly, we have shown that the costs for the use of public blockchains (from the Bitcoin family) are very reasonable. It is true that the payment per publication in these blockchains fluctuates according to the price of the associated cryptocurrency. In this way, we have seen that at times of high price ("speculation"), the cost may increase. It is also true that the signatories can establish a limit above which they are not willing to pay. In addition, an analysis conducted to cover a long period of time for different cryptocurrencies has allowed us to show that the maximum costs actually move in margins that should not be inconvenient for users.

Thirdly, we have demonstrated that our proposal is viable from the practical point of view. Through the implementation of a POC, we have been able to verify that it is possible to put into operation a contract signing application based on the blockchain, with a reasonable time cost of development. We can highlight the set of tools available to facilitate the implementation task.

Fourthly, our proposed contract signing protocol is not limited to the use of a specific blockchain. As we have shown, Ethereum can be considered as an alternative to blockchains based on Bitcoin for publishing the evidence of contract signing applications. Therefore, any blockchain allowing data to be published at a reasonable cost, can be easily integrated in our proposal.

Finally, we have compared our solution with the existing proposals for contract signing based on blockchain. The results, based on an analysis of efficiency, security and cost, prove that our proposal improves the previous solutions.

10.2. Industry and contextual applications

Any contract signature that has traditionally been made on paper between two parties is likely to benefit from the advantages of an electronic signature for that contract with the proposal we have made in this article. These advantages include greater agility (less time and more convenience in the signing process) and fewer errors (loss of copies, etc.).

As one of the parties must use cryptocurrencies of the blockchain, it would probably be advisable that the role of *Alice* is assumed by a merchant, which generally have greater means than a customer. This does not restrict the applicability between individuals, since many citizens have cryptocurrencies of different blockchains (or would be willing to obtain them). In any case, from our point of view, the scenarios where the solution is easily applied are business-to-business and business-to-administration, especially this latter because of the special security requirements that are usually demanded in the public procurement processes.

10.3. Limitations

We would highlight two limitations to the extensive implementation of the presented solution. First, we have the fact that one of the users must have cryptocurrency from the blockchain to support the protocol. This means that the user must buy coins of this cryptocurrency to pay for the publication costs of the transaction. In addition, these expenses are not fixed, in a double sense. On the one hand, they depend on a compromise between the desired speed of publication and the volume of pending transactions to be published. On the other hand, the cost depends on the price of the cryptocurrency at the time (more or less) of the publication (and certain cryptocurrencies have shown considerable fluctuations over time). Therefore, this is an issue that can affect any deployed application on a public blockchain.

The second limitation refers to the temporal aspect. To guarantee the requirement of timeliness (and, indirectly, that of fairness), we have established a deadline in the contract signing protocol, and indeed, we have shown that the protocol meets the requirements of timeliness and fairness. But the signatories do not have control over the amount of time that the miners may take to publish the transaction in the blockchain. In periods of high traffic (many transactions to be published), the deadline may expire even if the two signers have acted diligently. This implies that if they want to sign the contract, they will have to restart the execution of the protocol.

The solution to the two previous problems could come from the use of blockchains not linked to a cryptocurrency that also establish clear time limits for the publication of transactions. Possibly, this would be permissioned blockchains, which would open other possible problems (who manages the blockchain, what confidence it generates, etc.), which should be addressed in future work.

Acknowledgment

This work is partially financed by the European Social Fund and the Spanish Government under projects TIN2014-54945-R and RTI2018-097763-B-I00.

References

Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, Ł., 2014. Fair two-party computations via bitcoin deposits. In: International Conference on Financial Cryptography and Data Security. Springer, pp. 105–121.

Asokan, N., Schunter, M., Waidner, M., 1997. Optimistic protocols for fair exchange. In: Proceedings of the 4th ACM conference on Computer and communications security ACM, pp. 7–17.

Asokan, N., Shoup, V., Waidner, M., 1998. Asynchronous protocols for optimistic fair exchange. In: Proceedings. 1998 IEEE Symposium on Security and Privacy IEEE, pp. 86–99.

Bao, F., Wang, G., Zhou, J., Zhu, H., 2004. Analysis and improvement of micali's fair contract signing protocol. In: Australasian Conference on Information Security and Privacy. Springer, pp. 176–187.

Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.L., 1990. A fair protocol for signing contracts. IEEE Trans. Inform. Theory 36 (1), 40–46.

Bentov, I., Kumaresan, R., 2014. How to use bitcoin to design fair protocols. In: International Cryptology Conference. Springer, pp. 421–439.

Bitcoin-Magazine, 2017. The Long Road to SegWit: How Bitcoin's Biggest Protocol Upgrade Became Reality.https://bitcoinmagazine.com/articles/long-road-segwit-how-bitcoins-biggest-protocol-upgrade-became-reality/.

Bitcoin Testnet3 free faucet, 2019. Bitcoin Testnet3 free faucet. https://coinfaucet.eu/en/btc-testnet/.

BitcoinCore, 2019. Segregated Witness Wallet Development Guide. https://bitcoincore.org/en/segwit_wallet_dev/.

Bitcoinj, 2019. Bitcoin Java and Javascript library. https://bitcoinj.github.io/.

Blockchyper, 2019. Blockchyper Blockchain online explorer. https://www.blockcypher.com/.

Coin-Metrics, 2019. Open source cryptoasset analytics. https://coinmetrics.io/.

Conti, M., Kumar, E.S., Lal, C., Ruj, S., 2018. A survey on security and privacy issues of bitcoin. IEEE Commun. Surveys Tutorials 20, 3416–3452.

Damgård, I.B., 1995. Practical and provably secure release of a secret and exchange of signatures. J. Cryptol. 8 (4), 201–222.

Delgado-Segura, S., Pérez-Sola, C., Navarro-Arribas, G., Herrera-Joancomartí, J., 2017. A fair protocol for data trading based on bitcoin transactions. Future Generation Computer Syst.

Douceur, J.R., 2002. The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (Eds.), Peer-to-Peer Systems. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 251–260.

Etherscan, 2019. Block Explorer and Analytics Platform for Ethereum. https://etherscan.io/.

Even, S., Goldreich, O., Lempel, A., 1985. A randomized protocol for signing contracts. Commun. ACM 28 (6), 637–647.

Ferrer-Gomila, J.L., Payeras-Capella, M., Huguet-Rotger, L., 2004. Optimality in asynchronous contract signing protocols. In: International Conference on Trust, Privacy and Security in Digital Business. Springer, pp. 200–208.

Garay, J.A., Jakobsson, M., MacKenzie, P., 1999. Abuse-free optimistic contract signing. In: Annual International Cryptology Conference. Springer, pp. 449–466.

Goldreich, O., 1984. A simple protocol for signing contracts. In: Advances in Cryptology Springer, pp. 133–136.

Hasan, H.R., Salah, K., 2018. Blockchain-based proof of delivery of physical assets with single and multiple transporters. IEEE Access 6, 46781–46793.

Hasan, H.R., Salah, K., 2018. Proof of delivery of digital assets using blockchain and smart contracts. IEEE Access 6, 65439–65448.

Hawlitschek, F., Notheisen, B., Teubner, T., 2018. The limits of trust-free systems: a literature review on blockchain technology and trust in the sharing economy. Electron. Commerce Res. Appl. 29, 50–63.

Heilman, E., Baldimtsi, F., Goldberg, S., 2016. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In: International Conference on Financial Cryptography and Data Security. Springer, pp. 43–60.

Heilman, E., Kendler, A., Zohar, A., Goldberg, S., 2015. Eclipse attacks on bitcoin's peer-to-peer network. In: 24th USENIX Security Symposium. USENIX Association, Washington, D.C., pp. 129–144.

Huang, H., Li, K.-C., Chen, X., 2017. A fair three-party contract singing protocol based on blockchain. In: International Symposium on Cyberspace Safety and Security. Springer, pp. 72–85.

Jayasinghe, D., Markantonakis, K., Mayes, K., 2014. Optimistic fair-exchange with anonymity for bitcoin users. In: 2014 IEEE 11th International Conference on e-Business Engineering (ICEBE) IEEE, pp. 44–51.

Liu, J., Li, W., Karame, G.O., Asokan, N., 2016. Towards fairness of cryptocurrency payments. arXiv preprint arXiv:1609.07256.

Liu, J., Li, W., Karame, G.O., Asokan, N., 2018. Toward fairness of cryptocurrency payments. IEEE Security Privacy 16 (3), 81–89.

Mai, F., Shan, Z., Bai, Q., Wang, X., Chiang, R.H., 2018. How does social media impact bitcoin value? a test of the silent majority hypothesis. J. Manage. Inform. Syst. 35 (1), 19–52.

Nakamoto, S., 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.

Salah, K., Rehman, M.H.U., Nizamuddin, N., Al-Fuqaha, A., 2019. Blockchain for ai: review and open research challenges. IEEE Access 7, 10127–10149.

Tian, H., He, J., Fu, L., 2017. Contract coin: toward practical contract signing on blockchain. In: International Conference on Information Security Practice and Experience. Springer, pp. 43–61.

Wood, G., 2017. Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd - 2017-08-07). https://ethereum.github.io/yellowpaper/paper.pdf.

Zhang, Y., Deng, R.H., Liu, X., Zheng, D., 2018. Blockchain based efficient and robust fair payment for outsourcing services in cloud computing. Inform. Sci.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., Wang, H., 2018. Blockchain challenges and opportunities: a survey. Int. J. Web Grid Services 14 (4), 352–375.