Blockchain-Based Fair Certified Notifications

Macià Mut-Puigserver(✉), M. Magdalena Payeras-Capellà, and Miquel A. Cabot-Nadal

Dpt. de Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, Ctra. de Valldemossa, km 7,5., 07122 Palma, Spain
{macia.mut,mpayeras,miquel.cabot}@uib.es

Abstract. Lots of traditional applications can be redefined thanks to the benefits of Blockchain technologies. One of these services is the provision of fair certified notifications. Certified notifications is one of the applications that require a fair exchange of values: a message and a non-repudiation of origin proof in exchange for a non-repudiation of reception evidence. To the best of our knowledge, this paper presents the first blockchain-based certified notification system. We propose two solutions that allow sending certified notifications when confidentiality is required or when it is necessary to register the content of the notification, respectively. First, we present a protocol for Non Confidential Fair Certified Notifications that satisfies the properties of strong fairness and transferability of the proofs thanks to the use of a smart contract and without the need of a Trusted Third Party. Then, we also present a DApp for Confidential Certified Notifications with a smart contract that allows a timeliness optimistic exchange of values with a stateless Trusted Third Party.

Keywords: Blockchain · Fair certified notifications · Smart contract Confidentiality · Fairness · Cryptocurrencies · Certified electronic mail

1 Introduction

Blockchain technology provides an unalterable system of data registry that enables new solutions for a wide range of traditional applications. One of these traditional services that could benefit of the distinctive features of blockchain is the provision of certified notifications, that is, a service that allows a sender to prove that she has sent a message to a receiver or set of receivers. Thus, certified notification services provide evidence that a receiver has access to a message since a specific date/time. Certified notifications, along with other electronic services, such as electronic signature of contracts, electronic purchase (payment in exchange for a receipt or digital product) or certified mail, require a fair exchange of items between two or more users. In order to create protocols that allow carrying out these exchanges and, at the same time, maintain the security

J. Garcia-Alfaro et al. (Eds.): DPM 2018/CBT 2018, LNCS 11025, pp. 20–37, 2018.
https://doi.org/10.1007/978-3-030-00305-0_2

of communications, there are solutions that fall into the generic pattern named *fair exchange of values*. A fair exchange always provides an equal treatment of all users, and, at the end of each execution, either each party has the element she wishes to obtain from the other party, or the exchange has not been carried out successfully for no one (any party has received the expected item). In a typical notification case, the element to be exchanged is the message along with non-repudiation proof of origin and reception.

Fair exchange protocols proposed so far usually use TTPs [12,13,20], which are responsible for resolving any conflict that arises as a result of interrupted exchanges or fraud attempts. In addition to that, these protocols normally use non-repudiation mechanisms in order to generate evidence that proves the behavior of the actors of the protocol. Currently, with the advent of the blockchain technology and smart contracts, TTPs can be replaced or complemented by this new know-how, which opens a range of new possibilities to find effective solutions to the electronic versions of the protocols that fulfil the generic pattern of fair exchange of values. A method for designing new fair exchanges by means of the Bitcoin network is to motivate parties to complete the protocol in order to assure fairness by using a bond or a monetary penalty for dishonest parties [4,10,15].

In addition to that, the Ethereum blockchain and its cryptocurrency Ether offers an even richer functionality set than conventional cryptocurrencies such as bitcoin, since they support smart contracts in a fully distributed system that could lead, as we will see in this paper, to enable fair exchanges of tokens without the involvement of a TTP (since smart contracts are self-applied and reduce the need for trusted intermediaries or reputation systems that decrease transaction risks). This new technology allows us to define transactions with predetermined rules (written in a contract) in a programmable logic that can guarantee a fair exchange between parties with an initial mutual distrust. This feature prevents parties from cheating each other by aborting the exchange protocol and discharges the need for intermediaries with the consequent reduction of delays and commissions for their services.

The revealing power of the blockchain is further enhanced by the fact that blockchains naturally incorporate a discrete notion of time, a clock that increases each time a new block is added. The existence of a trusted clock/register is crucial to achieve the property of fairness in the protocols. This feature can make the cryptography model in the blockchain even more powerful than the traditional model without a blockchain where the fairness is very difficult to guarantee without the intervention of a TTP.

This paper aims to show how the blockchain technology and the smart contracts can introduce a new paradigm to deal with the fair exchange problem. By using this technology, we can reduce or even remove the role of the TTPs inside such protocols. As far as we know, there are no previous works that deals with blockchain-based fair certified notifications and smart contracts. Previous studies on fairness using blockchain focus on fair purchase operations between a product (or a receipt) in exchange of cryptocurrencies (usually bitcoin) [3,5,6,11].

[16] uses a smart contract for the resolution of a purchase operation while [1] uses smart contracts and trusted execution environments to guarantee the fair exchange of a payment and the result of an execution.

We present two Blockchain-based Systems for Fair Certified Notifications, the first proposal allows a non-confidential fair exchange of a notification message for a non-repudiation of reception token with no involvement of any TTP. The second one allows a confidential fair exchange of a notification for a non-repudiation of reception token. It has the optimistic intervention of a stateless TTP.

2 Ideal Properties of a Fair Certified Notification System.

Some requirements for fair exchange were stated in [2], and re-formulated in [21]:

1. **Effectiveness.** If two parties behave correctly, the exchange will take place and all parties will receive the expected items.
2. **Fairness.** After completion of a protocol run, either each party receives the expected item or neither party receives any useful information about the others item. The fairness is *weak* if, by the end of the execution, both parties have received the expected items or if one entity receives the expected item and another entity does not, the latter can get evidence of this situation.
3. **Timeliness.** At any time, during a protocol run, each party can unilaterally choose to terminate the protocol without losing fairness.
4. **Non-repudiation.** If an item has been sent from party A to party B, A can not deny origin of the item and B can not deny receipt of the item.
5. **Verifiability of Third Party.** If the third party misbehaves, resulting in the loss of fairness for a party, the victim can prove this fact in a dispute.

We can add, to this set of properties, some other interesting properties for the case of certified notifications.

6. **Confidentiality.** Only the sender and the receiver of the notification know the contents of the certified message.
7. **Efficiency.** An efficient protocol uses the minimum number of steps that allow the effective exchange or the minimum cost.
8. **Transferability of evidence.** The proofs generated by the system can be transferred to external entities to prove the result of the exchange.
9. **State Storage.** If the TTP that can be involved in the exchange is not required to maintain state information then the system is *stateless*.

Some of the above properties cannot be achieved in the same protocol. The authors of [9] enumerate the incompatibilities among the ideal features. Some examples are Weak Fairness and Transferability of Evidence, Stateless TTP and timeliness and Verifiability and transparenvy of the TTP. in this paper we will see that in a protocol that offers weak fairness a party cannot transfer the evidence to an arbiter since the other party could have contradictory evidence.

3 State of the Art of Fair Certified Notification Protocols

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

To overcome reluctances between the parties and to assure fairness, almost all the existing proposals use a TTP. This trusted third party can play and important role, participating in each exchange or a more relaxed role in which the TTP is only active in case of arising a dispute between the parties (optimistic protocols).

Due to the incompatibility among some of the properties and the difficulty to achieve simultaneously some other properties, we can find protocols that solve the exchange in an efficient way with an optimistic TTP although achieving only weak fairness [7], other systems focused in the achievement of specific features as the transferability of evidences [14], the verifiability of the TTP [17], the avoidance of the selective rejection based on the identity of the sender [19], the flexibility to allow the delivery to multiple receivers [8] or the reduction of the volume of state information that the TTP must maintain [18].

This paper proposes the use of blockchain-based technologies and the Ethereum ecosystem to implement a DApp for certified notifications in a decentralized way. This proposal does not require the use of electronic mail and, depending on the desired application, it does not requiere neither the use of a TTP to guarantee the fairness of the exchange. As far as we know, there are no proposals to solve the problem of certified notification using blockchain-based technologies. Only some papers about fair payments, very recently, refers to fair exchange protocols over blockchain [5,16].

4 Conceptual Design of Two Blockchain-Based Systems for Fair Certified Notifications

In this section we will analyse the possibilities of the use of blockchain-based technologies to provide a DApp for fair certified notifications reducing the involvement of trusted third parties compared with traditional approaches. We present a high level description of two solutions (the details of them will be presented in Sects. 6 and 7, respectively). One of them is well suited for those notifications that do not require the confidentiality of the message (or even it is required that the message can be public and accessible to everybody). The other solution allows the message to be hidden to others than the receiver. As it is showed in the descriptions, in the first approach there is no need of a TTP in any step of

the exchange nor in a dispute resolution phase while in the second proposal the TTP will be involved only in the dispute resolution phase (optimistic protocol). Moreover, it is not required that this TTP stores information of the state of any transaction.

4.1 Non-confidential Notifications Without TTP

In this first proposal we consider that confidentiality is not required or even not desired. The sender executes the first step of the protocol using the DApp to register the hash of the notification message on the blockchain. At this point, the receiver does not have access to the message, although the transaction remains stored in the blockchain due to the fact that the registered value is the hash of the message and not the message itself.

The sender will make a new transaction including the message in a third step, provided that the receiver would have made a previous transaction manifesting his desire to receive the notification.

The protocol for non-confidential certified notifications works as follows:

1. The sender, originator of the message, uses the smart contract to publish in the blockchain the hash of the notification message. Other parameters of this transaction are the address of the receiver and the deadline for the notification to be completed. Moreover, a deposit can be required in this step. The amount will be included in the transaction.
2. The receiver, if he accepts the reception of the notification, publishes a message expressing his will.
3. Finally, before the expiration of the deadline, the sender can execute the *finish* procedure to publish the message. As a consequence, the smart contract publishes the non-repudiation proof. If the execution of the exchange requires a deposit, the smart contract returns the amount to the sender.

   After the deadline, if the three steps have not been executed properly, the state of the exchange is not *finish* and then both parties can access a function in the smart contract to request the cancellation of the transferred elements.
   (a) Cancellation of reception, requested by the receiver if the sender does not publish the message when the receiver has accepted the notification.
   (b) Cancellation of delivery, requested by the sender, if the receiver has not accepted the notification.

   In both cases, the smart contract checks the identity of the user and the deadline. The smart contract generates a transaction to point out that the notification has been cancelled. In the first case, the sender will not receive the refund of the deposit (this way, the deposit is useful to motivate the sender to finish the exchange before the deadline).

Since the message is included in a transaction, it will be registered in the blockchain, so the notification in this case is not confidential. This protocol is executed entirely over Ethereum, so no off-chain communication between the parties is required. This way, there is no need of communication channels between the parties.

4.2 Optimistic Confidential Notifications with Stateless TTP

This second proposal has been designed taking into account those notifications that require confidentiality. That is, the blockchain has to preserve the fairness of the exchange but the message cannot be stored in a publicly accessible block. The main difference with the first proposal is that in this case the protocol allows an optimistic exchange, that is, the exchange can be executed completely without the intervention of the TTP nor the blockchain. Another important feature is that this proposal does not require a deadline and can be finished at any moment. A stateless TTP can be used to resolve the disputes that can arise between the parties.

The proposed protocol for confidential fair certified notifications is based in the protocol described in [7], an optimistic protocol in three steps with a trusted third party that is involved only in case of disputes between the parties. In [7] both parties can contact the TTP who maintains state information. The three steps of [7] are:

1. The sender A encrypts the message M with a simmetric key K, producing a ciphertext c. The key K is encripted with the public key of the TTP (it means that the TTP, who knows the correspondent private key, can decrypt it), producing K_t. A third element h_A is the signature of A on the concatenation of the hash of ciphertext c and K_t, part of the evidence of Non-Repudiation of Origin for B. Then A sends the triplet c, k_t and h_A.
2. B sends h_B, a signature of B on the concatenation of the hash of ciphertext c and k_t, evidence of Non-Repudiation of Receipt for B, to A.
3. A sends k_A, the key K enciphered with the private key of A, second part of the Non-Repudiation of Origin evidence for B.

During this three steps protocol the parties exchange the Non-Repudiation proofs together with elements that are useful in case of interruption of the exchange. These elements, as K_t, are managed by the TTP during a dispute resolution subprotocol. The execution of the dispute resolution subprotocol can be requested by both A and B contacting the TTP.

In this new blockchain-based solution, the originator A and the recipient B will exchange messages and non-repudiation evidences directly. Only as a last resort, in the case they cannot get the expected items from the other party, the smart contract or the TTP would be invoked, by initiating the *cancel* or *finish* functions. In comparison with the protocol described in [7], in the blockchain-based solution the role of the TTP has been reduced. The sender will never contact the TTP. The TTP will answer only requests from the receiver B by accessing to the smart contract that has been deployed. The TTP is totally stateless and, in any case, it stores information about the state of any exchange.

The protocol for confidential certified notifications works as follows: The parties, A (the sender) and B (the receiver) will execute a direct exchange in three steps, using the DApp (the details of it will be presented in Sect. 7).

1. A sends an encrypted message to B using a session key. Moreover, A also sends an element to B that could be useful in case of dispute, that is, if A

does not follow the steps of the protocol (i.e.: the session key encrypted with the public key of the TTP). The TTP is the responsible of the deployment of the smart contract that will manage the exchange.
2. B sends the non-repudiation proof.
3. A sends the key to decipher the message.
   If some party does not follow the protocol, the exchange can be resolved as follows:
   (a) If A does not receive the element of step 2., she can send a request to the smart contract. If the state of the notification is 'Created' (nor 'Cancelled' neither 'Finished'), then the state will be changed to 'Cancelled', indicating that the notification has not been performed successfully.
   (b) If B does not receive the message in step 3, B will contact the TTP providing the received elements in step 1 together with the non-repudiation of reception proof. The TTP will access the smart contract to check the state of the notification. If the notification has not been cancelled, the TTP will publish in the blockchain a non-repudiation of reception proof and the required elements for B to obtain the confidential message.

5 Smart Contracts Development Settings

In order to deploy smart contracts on the blockchain there are some frameworks that help the developer manage the deployment on the network. To develop the DApp we have interacted with the console of the Node JS platform[1]. This modular platform provides us with the necessary components to develop, test and deploy decentralized applications such as smart contracts. Functionalities provided by Node JS are implemented by independent modules and packages. In this way, we use NPM (Node Package Manager[2]) to easily install/uninstall, configure and update the different modules and software packages of the platform (called third-party modules).

An important configuration file is *package.json*. This is a JSON format file that is stored in the application root folder. This file provides the specific aspects to manage the module dependencies that the application requires. For instance, the file states information like our application name, module versions (name and version together work as an identifier that is assumed to be unique), license, directories, version control repository and so on. Keeping in mind this structure, a smart contract ready to deploy will be stored inside a folder within the solidity file that specifies the code of our application, the *package.json* and the node modules folder with all the necessary packages. Inside the root directory we have two more significant files: *compile.js* and *deploy.js*. Both are javascript files that, while compile.js specifies the requirements to compile the smart contract and the statement to compile it, the *deploy.js* file defines the tools used to deploy the smart contract. In order to deploy the smart contract, we need to have a connection to the Ethereum blockchain and to sign a transaction.

[1] https://nodejs.org/en.
[2] https://www.npmjs.com/.

We have deployed the smart contract on the Rinkeby testnet[3]. Rinkeby is the main Ethereum blockchain testnet that behaves similarly to the real Ethereum blockchain. We can acquire Rinkeby Ether for our account from a faucet[4]. Of course, before acquiring Ether we need to have an account that we can manage with an Ethereum wallet. For this reason, we have installed the Metamask[5] add-on in our browser. Metamask is an Ethereum browser that allows us to interact with the blockchain and to run Ethereumm DApps. Metamask also implements an Ether wallet that enables to sign blockchain transactions.

Therefore, the deploy.js file makes a call to the compile.js file to compile the contract and then specifies the web3 connection (i.e. the bridge to the blockchain) and the account needed to sign the transaction in order to deploy the contract. For this reason, we introduced the *HDWalletProvider* from Truffle[6] to handle the connection to the Ethereum network and sign the transaction. We have to give our private key from the Metamask wallet to the *HDWalletProvider* so as to sign the transactions.

However, to avoid setting up our own blockchain node, we have used the Infura Ethereum node cluster[7]. This service enables us to run our transaction on the blockchain without needing to establish our own blockchain node. To work with this cloud-based Ethereum client we just need to sign up and then we are provided with a token that enables us to connect to the Ethereum network (with this service we can use the main Ethereum network, Rinkeby, Kovan or Ropsten). The result of the execution of the deploy.js file is the address of the new deployed smart contract.

At this point, we can easily interact with our deployed contract by using Remix[8]. Remix is a browser-based solidity compiler that also supports testing and debugging smart contracts. To run transactions on Remix we just have to select the environment (Web3 Provider), our Ethereum account and the address where the smart contract is deployed. The Metamask will always ask us confirmation before signing any transaction on the blockchain. Remix will show the transaction in pending mode until it is validated.

In addition to that, we also have tested our smart contracts locally. Ganache[9] implements a personal blockchain made by the Ethereum development of smart contracts that runs in local. Besides, Ganache also simulates an Ethereum Virtual Machine (EVM). After launching Ganache in our computer, a list with the private and public key from ten recently created accounts is reported at the node console in order to use them as test bank (each account has 100 ether in its balance). Now, Ganache is ready to be used to deploy, run and test smart contracts without using any public network. In Fig. 1 we have depicted our development configuration.

[3] https://www.rinkeby.io.
[4] https://faucet.rinkeby.io/.
[5] https://metamask.io.
[6] http://truffleframework.com.
[7] https://infura.io.
[8] https://remix.ethereum.org.
[9] http://truffleframework.com/ganache/.

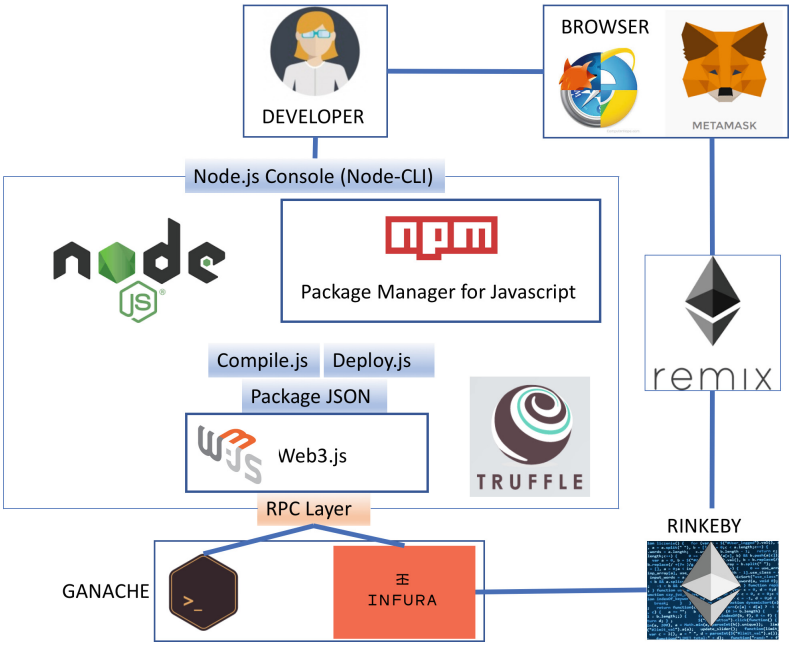

Fig. 1. Smart Contract Development Architecture

6 Development of the Non-Confidential Blockchain-Based Fair Certified Notifications Protocol Without TTP

6.1 Smart Contract

We have designed and implemented the smart contract for non-confidential notifications. For this proposal, a new instance of the smart contract will be created by the sender and it will manage all the steps of the exchange. It has been programmed using Solidity[10] and it has been deployed over an Ethereum network. Ethereum addresses have been assigned to both the sender A and the receiver B. Both A and B will interact with the blockchain using Web3.js interfaces (Fig. 4).

Figure 3 shows the smart contract that manages the non-confidential notifications. The constructor of the smart contract includes variables to store the addresses of the sender and the receiver, the hash of the notification message, the instant of execution of the first step of the exchange and the value of the execution period (the deadline). The contract includes five functions: to initiate the notification (the constructor), to accept a notification, to deliver the message, to cancel the exchange and to inquiry about the state of the exchange. The certified notification is created by the sender, who specifies the receiver, the hash of the message and the maximum duration of the exchange. The smart contract also has an attribute to store the content of the message.

[10] https://solidity.readthedocs.io/en/v0.4.21/.

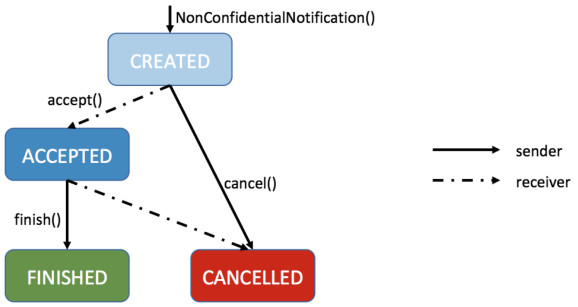

Fig. 2. Possible states of the exchange in the Non-Confidential Notifications Protocol.

```
pragma solidity ^0.4.24;

contract Notification {
    address public sender; // Parties involved
    address public receiver;
    bytes32 public messageHash; // Message
    string public message;
    uint public term; // Time limit (in seconds)
    uint public start; // Start time
    enum State {created, cancelled, accepted, finished } // Possible states
    State public state;

    event StateInfo( State state );

    constructor (address _sender, address _receiver, bytes32 _messageHash, uint _term)
    public payable {
        require (msg.value>0); // Requires sender deposits minimum 1 wei (>0 wei)
        sender = _sender;
        receiver = _receiver;
        messageHash = _messageHash;
        start = now; // now = block.timestamp
        term = _term;
        state = State.created;
        emit StateInfo(state);
    }
    function accept() public {
        require (msg.sender==receiver && state==State.created);
        state = State.accepted;
        emit StateInfo(state);
    }
    function finish(string _message) public {
        require(now < start+term); // It's not possible to finish after deadline
        require (msg.sender==sender && state==State.accepted);
        require (messageHash==keccak256(_message));
        message = _message;
        sender.transfer(this.balance); // Sender receives the refund of the deposit
        state = State.finished;
        emit StateInfo(state);
    }
    function cancel() public {
        require(now >= start+term); //  It's not possible to cancel before deadline
        require((msg.sender==sender && state==State.created) ||
                (msg.sender==receiver && state==State.accepted));
        if (msg.sender==sender && state==State.created) {
            sender.transfer(this.balance); // Sender receives the refund of the deposit
        }
        state = State.cancelled;
        emit StateInfo(state);
    }
}
```

Fig. 3. Smart Contract for the Non-Confidential Notifications Protocol.

The contract also manages a Solidity event to follow the progress of the exchange. This event *stateInfo* allows the parties to see the evolution of the state of the exchange (*created*, *accepted*, *finished* or *cancelled*).

Functions *Accept* and *Finish* check the identity of the address that throws the transaction. The address of the receiver and the sender are verified before updating the state. Function *Cancel* also checks the addresses. In this case, both sender and receiver can execute the function, depending on the state of the exchange. Moreover, all the functions that can cause any change in the state of the exchange check the value of the variable *State*. Function *Finish* requires that the present time does not exceed the deadline before executing its code. Also, function *Cancel* verifies that the current time is greater than the deadline before carrying on with the execution of it.

The smart contract will manage the publication on the blockchain of all the values of the required variables to maintain the fairness of the exchange following the protocol described in Sect. 4.1.

6.2 Properties

The non-confidential certified notifications protocol allows the fair exchange of a message and non-repudiation proofs. The main properties achieved by the protocol are analyzed in this section.

- **Strong Fairness.** *A* will not receive the non-repudiation proof of reception provided by the smart contract unless she executes the transaction to register the message in the blockchain (case *State=finished*). On the other hand, *B* will not have access to the message unless he executes the transaction to accept the notification (*State=Accepted*). At any moment, the smart contract does not generate alternative cancellation or finalization proofs that could create any situation where one of the parties can have contradictory proofs (leading the exchange to weak fairness), as can be seen in Fig. 3.
- **Total absence of TTP. Substitution by an smart contract.** This proposal does not require an external party acting as a TTP. The parties execute the functions of the smart contract creating the associate transactions and there is no need of dispute resolution.
- **Transferability of the proofs.** Since the parties cannot obtain contradictory proofs in any way, the generated proofs can be presented as evidence to an external entity. Moreover, its transferability is easy, since the results of the exchange are stored in the blockchain. Due to the immutability of the blockchain, the content of the notification cannot be modified so the system provides integrity to the notification. The moment that the notification takes place can be derived from the timestamp of the block where the transaction is included.
- **Weak Timeliness.** The protocol is not asynchronous. If one of the parties delays its intervention in the exchange, the other party will not be able to resolve it until the deadline. However, after the deadline both parties can request the finalization of the exchange. Moreover, the protocol wants to

motivate the sender to conclude the exchange before the timeout blocking an amount of money in the smart contract. This amount will only be refunded to the sender if she concludes before the deadline.

– **Non Repudiation.** The protocol achieves non-repudiation of origin together with non-repudiation of receipt after the execution of the exchange. A cannot deny having sent the message since there is a transaction on the blockchain from her address containing the message and another one related with the same message including the address of the receiver and the hash of the message. B cannot deny having received the notification since there is a transaction from his address in the blockchain accepting the reception of the message and the *State* of the exchange is *Finished*, so the message is publicly accessible in the blockchain.

7 Development of the Confidential Blockchain-Based Fair Certified Notifications Protocol

7.1 Smart Contract

We have designed and implemented a DApp that allows the optimistic exchange between the parties and a smart contract for the resolution of disputes. The smart contract has been programmed in Solidity and deployed over the Ethereum network (see Sect. 5). Ethereum addresses have been assigned to the sender A, the receiver B and the TTP. In comparison with the protocol described in [7], the role of the TTP has been reduced. The sender will never contact the TTP. The TTP will answer only requests from the receiver B by accessing to the smart contract that has been deployed. The TTP is totally stateless and, in any case, it stores information about the state of any exchange. Both A and B can interact with the blockchain if it is necessary through the Web3.js interface. For this reason, we have also designed a web service where the web client can connect using TLS protocol. This web service is used the off-chain communication exchanges between sender and recipient described in the protocol. In order to implement the cryptographic operations, we have used Stanford Javascript Crypto Library[11]. This enables us to use AES for the symmetric encryption operation, EC-ElGammal for the asymmetric encryption operations and ECDSA for the signature functions. However, the implementation of a PKI and the secure exchange of public keys are beyond the scope of this work. Figure 5 shows the smart contract that will manage the possible disputes between the parties after the execution of the exchange described in Sect. 4.2 for confidential notifications. This smart contract is deployed by the TTP, who defines the identities of the sender and the receiver. The smart contract manages the variable *state* in order to keep track of the state of each exchange (Fig. 2).

The event *stateInfo* allows the tracking of the evolution of each exchange state. The function *Cancel* checks the identity of the address that throws the transaction, which must be the address of the sender, together with the value

[11] http://bitwiseshiftleft.github.io/sjcl/.

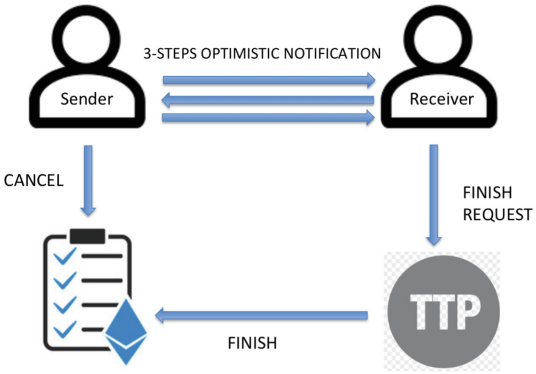

Fig. 4. Interaction between the actors.

of the variable *state*. This function can be executed only by the sender. The function *Finish* checks the identity of the party that sends the transaction, that is, the TTP, together with the value of the variable *state*. The TTP will execute this function if it receives a request from the receiver. The smart contract is responsible for the publication in the blockchain of the values of the elements used to maintain the fairness of the exchange following the protocol described in Sect. 4.2. In function *Finish*, if the conditions are fulfilled, the smart contract publishes in the blockchain both the non repudiation of reception proof (hB) and the session key encrypted with B's public key (hBt).

7.2 Properties

The main properties achieved by the confidential certified notifications protocol are analysed in this section.

- **Weak Fairness.** The protocol does not allow that any of the parties receive the expected item if the other party does not receive it. However, the intervenction of the TTP can lead to a situation in where one of the parties possesses contradictory evidence. A malicious A can have the non-repudiation proof received directly from B and also the cancellation proof generated by the smart contract after a cancellation request from A. For these reason, the fairness will be weak and the generated proofs are non transferable. Comparing this feature with the version of the protocol without blockchain, this protocol does not require that the arbitrator consults both parties to resolve the final state of the exchange. It is enough to check one of the parties and then match this version with the contents of the blockchain.
- **Optimistic.** The parties can finalize the exchange without the need to contact with a TTP or execute any function of the smart contract. If the parties do not follow the protocol and the execution of the smart contract is required, the gas necessary for its operation would be reduced compared with the protocol for non-confidential notifications protocol.

```
pragma solidity ^0.4.11;

contract ConfidentialNotifications {

   //Parties involved
    address sender;
    address receiver;
    address ttp;

    string hB; //NRR proof
    string hBt; //Intervention proof

    //Possible states
    enum State { created, cancelled, finished }
    State public state;

    function ConfidentialNotification (address _sender, address _receiver){
        ttp = msg.sender;
        sender = _sender;
        receiver = _receiver;
        state = State.created;
        }


    event stateInfo(
        State state
        );

    function Cancel() returns (string) {
         if(msg.sender==sender){
             if(state==State.created){
                 state=State.cancelled;
                 //return abort token
                 stateInfo(state);
             }else if (state == State.finished){
                 return hB;
             }
         }
    }

    function Finish(string _hB, string _hBt) returns (State) {
        if (msg.sender==ttp){
            if(state==State.cancelled){
                return state;
            }else{
                hB=_hB;
                hBt=_hBt;
                state=State.finished;
            }
        }
    }

    function getState() returns (string){
        if(state==State.cancelled) return "cancelled";
        if(state==State.created) return "created";
        if(state==State.finished) return "finished";
    }
}
```

Fig. 5. Smart Contract for Confidential Notifications.

- **Stateless TTP.** When the TTP is involved in the exchange, it can resolve the exchange through the use of the smart contract. The TTP does not need to store any kind of state information of the exchange.
- **Timeliness.** The parties can finish the exchange at any moment accessing the smart contract (sender A) or contacting the TTP (receiver B). The duration of the resolution will depend of the block notification treatment. The protocol can assume that the transactions are valid immediately (zero confirmation) or wait until the block is confirmed in the chain (fully confirmation).
- **Non repudiation.** The protocol achieves non-repudiation of origin together with non-repudiation of receipt after the execution of the three step exchange

or the finalization using the smart contract. A cannot deny having sent the message since B has the element received in the third step or the state of the smart contract is *Finished*. B cannot deny having received the notification since A has the elements sent by B in the second step of the protocol.

– **Confidentiality.** If the exchange is finished through the execution of the three step exchange protocol, then no other entity is involved in the exchange, and the message remains confidential. If the TTP is involved or the functions of the smart contract are executed, then the TTP will process the received elements and will make a transaction including the element that will allow B to decrypt the message but the plain message is not included in the transaction so it will not be included in a block of the blockchain to preserve the confidentiality.

8 Comparison and Conclusions

Previous solutions for fair certified notifications are mainly based on the intervention of a TTP that acts as an intermediary between sender and receiver. In this model of fair exchange, both parties obtain the expected item from the other or neither obtains what was expected. That is, either the issuer has received a non-repudiation of reception evidence and the recipient has received the message, or neither party obtains the desired item, the TTP can intervene to guarantee the fairness of the exchange if some participant misbehaves.

This paper presents two alternatives for sending certified notifications on a blockchain-based fairness. On the one hand, the first solution (see Sect. 6) allows users to send non-confidential notifications, the new DApp supports the sending and receiving of certified messages and guarantees the fairness of the exchange without requiring the intervention of any TTP to guarantee the security properties of the exchange since the actions of the different actors are recorded in the blockchain and, in the event that any actor does not fulfil the protocol, the smart contract will generate the corresponding evidence to preserve fairness. This proposal also preserves the properties of limited Timeliness (involved parties can be certain that the protocol will be completed at a certain finite point in time [16]), Transferability of proofs and Non-repudiation as it is stated in Sect. 6.2.

On the other hand, the second solution (see Sect. 7) is a fair exchange protocol that allows users to send confidential notifications and introduce an optimistic TTP (its intervention is only required if a party does not fulfil the protocol) to guarantee fairness. Thanks to the usage of a blockchain and a smart contract, the TTP can be stateless (i.e. the TTP does not need to store the state of the exchange regarding any protocol execution because all the information of each exchange is stored in the blockchain by using the smart contract). This solution assures the fair exchange (weak fairness, see Sect. 7.2). Like the first solution, this proposal also preserves Timeliness and Non-repudiation properties. However, Transferability of proofs is not strictly provided because anyone who want to verify the correctness of the exchange not only has to check the provided evidence by the parties but also has to check the blockchain. Table 1 compares

Table 1. Comparasison of Properties

Property	Non Confidential Notifications	Confidential Notifications
Non-repudiation	YES	YES
Fairness	STRONG	WEAK
Timeliness	LIMITED	YES
Effectiveness	YES	YES
TTP	NO	OPTIMISTIC/STATELESS
Evidence Transferibility	YES	NO
Confidentiality	NO	YES

Table 2. GAS per execution of the Non-Confidential Contract.

	Non-Confidential Notifications
Deployment	1086913
Accept	43644
Finish	59835
Cancel (created)	53011
Cancel (accepted)	30443

Table 3. GAS per execution of the Confidential Contract.

	Confidential notifications
Deployment	800433
Finish (Cancelled)	26388
Finish	88387
Cancel	44698
Cancel (Finished)	24772

the properties of both solutions while Tables 2 and 3 present the gas required for the execution of each function, for both protocols.

9 Future Work

There are some points to be studied to improve the proposed protocols. Thus, as further works we are going to:

- Test the smart contracts on real-like networks, checking confirmation delay time of transactions and possible undesired effects caused by these delays. Also we would like to obtain and evaluate an accurate register of the performance of our smart contracts in these real-like scenarios.
- Modify smart contracts to allow them to manage more than just one notification. This can be done with a contract that can create notification structs or

other new contracts that represent a notification exchange, and storing them into an array.
- Modify the confidential notification smart contract so that the TTP can create a main contract, and from this, multiples notifications can be created by the senders.
- Improve the use of events. Users can receive notifications, but they need the address of the smart contract in order to be subscribed to them. For this reason, the subscription to events can be also improved using only one smart contract that manages multiple notifications.
- In Tables 2 and 3 we have evaluated the necessary amount of gas to execute the contracts. However, we have left for further works a deeper analysis in order to find possible improvements to the code so as to reduce the commissions paid for using the blockchain.

The systems presented in this article show different sets of properties, so the choice of one system will depend basically on the needs of each exchange. As a future work, it is proposed to reformulate the system of confidential notifications to achieve strong fairness. The use of the blockchain for the diffusion of each of the executed steps and the full confirmations will be the element used to obtain this property.

Acknowledgments. This work has been partially financed by AccessTur TIN2014-54945-R AEI/FEDER UE and the network Consolider ARES TIN2015-70054-REDC projects.

References

1. Al-Bassam, M., Sonnino, A., Król, M., Psaras, I.: Airtnt: Fair Exchange Payment for Outsourced Secure Enclave Computations, CoRR, volume abs/1805.06411 (2018)
2. Asokan, N., Shoup, V., Waidner, M.: Asynchronous protocols for optimistic fair exchange. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, pp. 86–99, Oakland, CA, May 1998
3. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to better — how to make bitcoin a better currency. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 399–414. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_29
4. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_24
5. Delgado-Segura, S., Perez-Sola, C., Navarro-Arribas, G., Herrera-Joancomarti, J.: A fair protocol for data trading based on Bitcoin transactions. Future Gener. Comput. Syst. (2017)
6. Ethan H., Baldimtsi, F., Alshenibr, L., Scafuro, A., Goldberg, S.: TumbleBit: An Untrusted Tumbler for Bitcoin-Compatible Anonymous Payments. In: Network and Distributed System Security Symposium (NDSS) (2017)
7. Lluís Ferrer-Gomila, J., Payeras-Capellà, M., Huguet i Rotger, L.: An efficient protocol for certified electronic mail. In: Goos, G., Hartmanis, J., van Leeuwen, J., Pieprzyk, J., Seberry, J., Okamoto, E. (eds.) ISW 2000. LNCS, vol. 1975, pp. 237–248. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44456-4_18

8. Ferrer-Gomila, J.L., Payeras-Capellá, M., Huguet-Rotger, L.: A realistic protocol for multi-party certified electronic mail. In: Chan, A.H., Gligor, V. (eds.) ISC 2002. LNCS, vol. 2433, pp. 210–219. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45811-5_16
9. Ferrer-Gomilla, J., Onieva, J., Payeras-Capellà, M., Lopez, J: Certified electronic mail: properties revisited. Comput. Secur. **29**(2), 167–179 (2010)
10. Goldfeder, S., Bonneau, J., Gennaro, R., Narayanan, A.: Escrow protocols for cryptocurrencies: how to buy physical goods using bitcoin. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 321–339. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_18
11. Heilman, E., Baldimtsi, F., Goldberg, S.: Blindly signed contracts: anonymous on-blockchain and off-blockchain bitcoin transactions. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 43–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_4
12. Huang, Q., Yang, G., Wong, D., Susilo, W.: A new efficient optimistic fair exchange protocol without random oracles. Int. J. Inf. Secur. **11**(1), 53–63 (2012). ISSN 1615–5270
13. Huang, Q., Wong, D.S., Susilo, W.: P^2OFE: privacy-preserving optimistic fair exchange of digital signatures. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 367–384. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_19
14. Kremer, S., Markowitch, O.: Selective receipt in certified e-mail. In: Rangan, C.P., Ding, C. (eds.) INDOCRYPT 2001. LNCS, vol. 2247, pp. 136–148. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45311-3_14
15. Küpçü, A., Lysyanskaya, A.: Usable optimistic fair exchange. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 252–267. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11925-5_18
16. Liu, J., Li, W., Karame, G., Asokan, N.: Towards Fairness of Cryptocurrency Payments. In: IEEE Security and Privacy (2017)
17. Mut-Puigserver, M., Ferrer-Gomila, J., Huguet-Rotger, L.: Certified e-mail protocol with verifiable third party. In: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, pp. 548–551 (2005)
18. Onieva, J., Zhou, J., Lopez, J.: Enhancing certified email service for timeliness and multicast. In: Fourth International Network Conference, pp. 327–335 (2004)
19. Payeras-Capellà, M., Mut-Puigserver, M., Ferrer-Gomila, J., Huguet-Rotger, L.: No Author Based Selective Receipt in an Efficient Certified E-mail Protocol. In: PDP 2009, pp. 387–392 (2009)
20. Shao, Z.: Fair exchange protocol of Schnorr signatures with semi-trusted adjudicator. Comput. Electric. Eng. (2010)
21. Zhou, J., Deng, R., Bao, F.: Some remarks on a fair exchange protocol. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 46–57. Springer, Heidelberg (2000). https://doi.org/10.1007/978-3-540-46588-1_4