Smart Contract for Multiparty Fair Certified Notifications

M. Magdalena Payeras-Capellà
Dpt. C. Matemàtiques i Informàtica
Universitat de les Illes Balears
Palma, Spain
mpayeras@uib.es

Macià Mut-Puigserver
Dpt. C. Matemàtiques i Informàtica
Universitat de les Illes Balears
Palma, Spain
macia.mut@uib.es

Miquel À. Cabot-Nadal
Dpt. C. Matemàtiques i Informàtica
Universitat de les Illes Balears
Palma, Spain
miquel.cabot@uib.es

***Abstract*—Blockchain can be the base for an evolution in several digital services. Fair Certified Notification is one of these and requires a fair exchange of values: a message and a non-repudiation of origin proof in exchange for a non-repudiation of reception evidence. The main contribution of this paper is the introduction of a blockchain-based Multiparty Fair Certified Notification system allowing the simultaneous fair notification to a set of receivers. We present and evaluate a smart contract for an optimistic multiparty protocol that achieves the properties of confidentiality, fairness and timeliness with a stateless TTP.**

***Index Terms*—Cryptographic Protocols, Blockchain, Certified Notifications, Smart Contracts, Confidentiality, Certified e-mail.**

I. Introduction

Trust, together with privacy and security, is an essential property of cryptographic protocols. Commonly, trust is assured by the involvement of Trusted Third Parties (TTP) but a smart and self-regulated management can avoid the use of this kind of intermediaries. The blockchain can facilitate self-organization by providing a self-management platform for companies, NGOs, foundations, government agencies, academics, and individual citizens. Parties can interact and exchange information on a global and transparent scale. After some years of evolution, blockchains are now capable of storing arbitrary data and establishing permissions to modify that data through self-administering and self-executing scripts which are performed by a distributed virtual machine. These scripts are known as smart contracts, and they allow platform operators to define complex and fully customizable rules which govern the blockchain's interaction with its users.

Blockchain technology provides an unalterable new digital ecosystem 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 or certified mail, require a fair exchange of items between two or more users.

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 wish 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 proofs of origin and reception.

Fair exchange protocols proposed in academic papers so far usually use TTPs [10], [11], [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. Also, the fairness of Certified Notifications commercial web-services[1] also rely on the use of TTPs.

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 built new cryptographic protocols and find effective solutions to the electronic versions of the protocols that fulfil the generic pattern of fair exchange of values.

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 and could 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 and discharges the need for intermediaries with the consequent reduction of delays and commissions for their services.

The paper presents a new Blockchain-based Protocol for Certified Notifications that allows a confidential fair exchange of notifications among a sender and multiple recipients. It

[1] https://www.safestamper.com/notify/
https://en.xolido.com/lang/xolidosign/xolidosignprofessional/
https://logalty.com/en/about-logalty/

978-1-5386-9184-7/18/$31.00 ©2018 IEEE
DOI 10.1109/CANDARW.2018.00089

has the optimistic intervention of a stateless TTP and the amount of trust that the user should deposit in the TTP has been minimized thanks to the intervention of the blockchain, which is publicly verifiable. As far as we know, there are no previous works that deals with blockchain multiparty certified notifications and smart contracts. We presented in [16] two blockchain-based proposals for certified notifications with a single receiver. Other studies on fairness using blockchain focus on fair purchase operations between a product (or a receipt) in exchange of cryptocurrencies (usually bitcoin) [2], [4], [5], [9], while [14] uses, for the first time, a smart contract for the resolution of a purchase operation.

II. Multiparty Fair Certified Notifications

In a multiparty scenario for Fair Certified Notifications, two or more parties agree to use a protocol for exchanging messages and collecting evidence of the transactions performed for the exchange of those messages. One of the users act as a sender, while the remaining users will be receivers of the same notification.

A. Ideal Properties

Some properties for multiparty fair certified notification protocols [1], [18], [21] are:

- **Effectiveness.** If the parties behave correctly, they will receive the expected items.
- **Fairness.** After completion of a protocol run, either each party receives the expected item or neither party receives any useful information about the other's item. The fairness is *weak* if, by the end of the execution, both parties have received the expected items or if one entity receives the expected item and another entity does not, the latter can get evidence of this situation. A multiparty protocol is said to be fair if, at the end of the protocol, all honest parties receive what they expect or none of them receive any valuable information.
- **Timeliness.** At any time, each party can unilaterally choose to terminate the protocol without losing fairness. A multiparty protocol is said to respect timeliness if all honest entities can terminate the protocol in a finite amount of time without losing fairness.
- **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.
- **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.
- **Confidentiality**. Only the sender and the receiver of the notification know the contents of the certified message. A multiparty protocol is said to be confidential if only the aimed honest recipients can disclose the message.
- **Efficiency.** An efficient protocol uses the minimum number of steps to allow an effective exchange or a minimum cost.
- **Transferability of evidences.** The proofs generated by the system can be transferred to external entities to prove the result of the exchange.
- **State Storage.** If the TTP has not to store information about the state of the exchange, then the system is *stateless*.

Some of the above properties cannot be achieved in the same protocol. The authors of [8] enumerate the incompatibilities among the ideal features. Some examples are Weak Fairness and Transferability of evidences, Stateless TTP and Asynchrony and Verifiability of the TTP and Transparency.

B. State of the Art of Certified Notification Protocols.

Certified notifications follow 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 shown in this paper. A certified notification includes an exchange of elements between the sender and the receiver or set of receivers; the sender has to send a message, then the receiver is able 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 [6], other systems focused in the achievement of specific features as the transferability of evidences [12], the verifiability of the TTP [15], the avoidance of the selective rejection based on the identity of the sender [19], the flexibility to allow the delivery to multiple receivers [7] or the reduction of the volume of state information that the TTP must store [17].

[18] analyses the properties of these exchanges in multiparty scenarios. In the same way as in the exchanges between two parties, the literature does not agree on the properties that the protocols must offer, which is why a compromise is made between properties, depending on the specific application. However, we can see a dominant trend in the use of TTPs in multiparty protocols, many of the proposals opt for optimistic approaches. [13] presents a protocol that uses group signatures with an on-line TTP, [7] presents an efficient multiparty optimistic protocol with the properties of asynchrony and verificability of the TTP while [22] describes a very efficient, asynchronous optimistic protocol.

This paper proposes the use of blockchain technologies and the Ethereum ecosystem to implement a DApp for certified notifications in a decentralized way. As far as we know, there are no proposals to solve the problem of multiparty certified notifications using blockchain-based technologies and only our previous work, [16], deals with blockchain-based certified notifications with a single receiver.

III. Blockchain-based Protocol for Multiparty Certified Notifications

In this section we present the protocol for blockchain-based multiparty certified notifications reducing the involvement of third parties compared with traditional approaches.

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.

TABLE I
Notation

Notation	
X, Y	Concatenation of messages X and Y
H(X)	Collision-resistant hash function applied to X
$PR_i[H(X)]$	Digital signature on the digest of message X
$i \rightarrow j$: X	Sending X from i to j
$i \Rightarrow j$: X	Broadcast X from i to all j's
$p \blacktriangleright e.f$	Execution of function f of e by p

TABLE II
Elements

Elements	
B	Set of Receivers.
B_i	An specific receiver.
B'	Subset of B with the users that will finish the exchange with A.
B''	Subset of B with the users that will not finish the optimistic exchange with A.
$B'' - finished$	Members of B that have contacted the TTP before the cancellation of the exchange.
$B'' - cancelled$	Members of B that have not contacted the TTP before the cancellation of the exchange.
SM	Smart Contract.
M	Certified Message to send from A to B.
$c = E_K(M)$	Symmetric cipher of message M with key K.
$k_T = PU_T(K)$	Key K encrypted with the public key of the TTP.
$h_A = PR_A[H[H(c), B, k_T, Id]]$	A's signature on the concatenation of the hash of c, B, k_T and Id. First part of the Non-Repudiation of Origin proof.
$h_{Bi} = PR_{Bi}[H[H(c), k_T, Id]]$	B's signature of the concatenation of the hash of c, k_T and Id. Non-Repudiation of receipt proof.
$k_A = PR_A[K, B', Id]$	A's signature on the key K together with B'. Second part of the Non-Repudiation of origin proof for B_i in case of belonging to B'.
$k'_T = PR_T[K, B_i, Id]$	TTP's signature on key k for user B_i. Alternative second part of the Non-Repudiation of origin proof.
$h_{BiT} = PR_B[H[H(c), k_T, h_A, h_{Bi}, Id]]$	Evidence of the TTP intervention requested by B_i.

The multiparty exchange protocol presents a solution for the fair certified notifications with a set of receivers. Thus, the protocol involves just one sender, but messages can be addressed to several recipients. We have designed the new protocol improving a non blockchain-based scheme that we presented in a previous work [7]. The notation used to describe the protocol is in Table I while the elements are listed in Table II. The multiparty protocol has three sub-protocols. Next, we explain each of these phases, as well as the values that must be exchanged by the different parties. The subprotocols are: *Optimistic Exchange*, *Cancel* and *Finish* Subprotocols.

A. Multiparty Optimistic Exchange Subprotocol

The protocol is optimistic in the sense that it is possible for a sender A to complete the exchange with the set of receivers B's without the intervention of the TTP. It is depicted in Subprotocol 1.

Subprotocol 1: Optimistic Exchange

1. $A \Rightarrow B : c, k_T, B, h_A$
2. $B_i \rightarrow A : h_{Bi}$, where $B_i \in B$ and $i \in \{1, ..., |B|\}$
3. $A \Rightarrow B'$: k_A

If the execution of these steps has been successfully completed, then each of the B's recipients obtains the key used to decipher the message k_A, as well as the corresponding NRO (Non-Repudiation of Origin) evidence (h_A). In the same way, the sender of the message will obtain the NRO evidence (h_{Bi}) from each recipient.

B. Multiparty Cancel Subprotocol

The Cancel subprotocol will be initiated by the sender of the message. The sender invokes the corresponding function of the Smart Contract in the case of not receiving the h_{Bi} parameter from each intended recipient. The Cancel subprotocol is illustrated in Subprotocol 2.

Subprotocol 2: Multiparty Cancel

1. A ▶ SM.cancel
2. SM: **FOR** ($\forall B_i \in B$")
   2.1.**IF**($B_i \in [B" - finished]$) **THEN**
   SM $\rightarrow$ A: h_{Bi}
   2.2.**IF**($B_i \notin [B" - finished]$) **THEN**
   Add B_i to $B" - cancelled$

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

C. Multiparty Finish Subprotocol

The Finish subprotocol will be initiated by any receiver, in the case of having sent the corresponding h_{Bi} but not having received the key k_A. This finalization will be carried out by the TTP based on the request received from the recipient B_i. After checking the correctness of the different parameters received from B_i the TTP invokes the *finish* function of the smart contract (the TTP submits the B_i's NRR evidence (h_{Bi}) to the smart contract). The procedure is illustrated in Subprotocol 3.

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

Subprotocol 3: Multiparty Finish

1. B_i ▶ TTP: $H(c), k_T, h_A, h_{Bi}, hBiT$
2. TTP ▶ SM.finish(h_{Bi})
   - 2.1. **IF**($B_i \in [B" - cancelled]$)
     - 2.1.1' SM → TTP: "$cancelled$"
   - 2.2. **ELSE**
     - 2.2.1. SM stores h_{Bi}
     - 2.2.2. SM adds B_i to $[B"finished]$
   - 3.1. **IF**(SM → TTP: "$cancelled$")
     - 3.1.1. TTP → B_i: $PR_T[H("cancelled"), h_{Bi}]$
   - 3.2. **ELSE**
     - 3.2.1. TTP → B_i: k'_T

IV. Development of the Protocol

In this section we present three smart contracts. The different smart contracts are useful to evaluate the performance of a multiparty notification on the blockchain both using a reusable two-party protocol and a multiparty protocol.

A. Smart Contract for a Two-Party Protocol

Smart Contract 1 manages the disputes between the parties after the execution of the exchange between two users.

```
pragma solidity ^0.4.11;

contract CertifiedMail {

address sender; //Parties involved
address receiver;
address ttp;
string hB; //NRR proof
string keyB; //Symmetric key encrypted with pubKey from B

enum State { created, cancelled, finished }//PossibleStates
State public state;
bytes15 stateString;

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

function cancel() {
if(msg.sender==sender){
if(state==State.created){
state=State.cancelled;
cancelEvent(getState());
}else if (state == State.finished){
cancelEvent(hB);}}}

function finish(string _hB, string _keyB){
if (msg.sender==ttp){
if(state==State.cancelled){
finishEvent(getState());
}else{
hB=_hB;
keyB=_keyB;
state=State.finished;
finishEvent(getState());}}}

...}
```

Smart Contract 1. Two-party Protocol

This first version of the *Smart Contract* allows the two-party fair certified notifications without the use of message identifiers. This means that each contract will be used only for an exchange and it 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.

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 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. 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).

B. Adding Identifiers to the Notifications

Smart Contract 2 will be used by the TTP to manage multiple exchanges involving different senders and receivers through the use of message identifiers. We can find a *struct* containing the identificative parameters of an exchange that will be stored in *maps*. Now, the sender and the receiver are not fixed during the creation of the contract but in the creation of the message (during *Cancel* or *Finish*).

```
pragma solidity ^0.4.11;

contract IDCertifiedMail {

enum State {cancelled, finished }
address ttp;
struct Message{
address sender;
address receiver;
string hB;
string keyB;
State state;
bool isValue;}
mapping(address => mapping(uint => Message)) messages;

constructor(){
ttp = msg.sender;}

function cancel(uint id,address receiver) {
if(messages[msg.sender][id].isValue){
if(messages[msg.sender][id].state==State.cancelled){
cancelEvent(stateToString(messages[msg.sender][id].state));
}else{cancelEvent(messages[msg.sender][id].hB);
}}else{createMessage(id,msg.sender,receiver, "",
"",State.cancelled);}}

function finish(uint id,address sender,address receiver,
string _hB, string _keyB){
if(msg.sender==ttp){
if(messages[sender][id].isValue){
finishEvent(stateToString(messages[sender][id].state));
}else{
createMessage(id,sender,receiver, _hB,
_keyB,State.finished);}}}

function createMessage(uint id, address sender, address
receiver, string _hB,string _keyB,State state) private {
messages[sender][id].sender=sender;
messages[sender][id].receiver=receiver;
messages[sender][id].hB=_hB;
messages[sender][id].keyB=_keyB;
messages[sender][id].state=state;
messages[sender][id].isValue=true;}

...}
```

Smart Contract 2. Reusable Protocol

Thanks to the use of a *map of maps*, an attacker will not be able to cancel a message that does not belong to him. Each

message also has a State, than can change its value with the execution of the functions *cancel()* and *finish()*.

The *Cancel* function checks if there exists a message with that identifier and sender. If it already exists, then the exchange has been previously cancelled or finished. If it is finished, the function tells it to the user, while if it is cancelled the smart contract gives him the NRR proof h_B. If the message does not exist then the message is created with state cancelled. The *Finish* function has to be executed by the TTP. It also checks if the message has already been created. If it is the case, it gives the value of state to the TTP. If the message has not been created previously, it creates the message using the identifier, the address of the sender and the parameters h_B and k ciphered with B's public key. The state of the exchange will be finished.

C. Smart Contract for the Multiparty Protocol

Smart Contract 3 is based on the previous one, but now it follows the Multiparty Certified Notifications Protocol explained in section § III. Now each message can be associated with multiple receivers.

```
pragma solidity ^0.4.11;

contract Multiparty {

enum State {cancelled, finished}
address ttp;
struct Receiver{
address receiver;
string hB;
string keyB;
State state;
bool isValue;}

struct Message{
uint id;
address sender;
mapping(address => Receiver) receivers; //All receivers
bool isValue;}
mapping(address => mapping(uint => Message)) messages;

constructor(){
ttp = msg.sender;}
...

function cancel (uint id,address[] cancelledReceivers) {
if (messages[msg.sender][id].isValue){
for (uint i = 0; i<cancelledReceivers.length;i++){
address receiverToCancel = cancelledReceivers[i];
if ((messages[msg.sender][id].receivers[receiverToCancel].
isValue)&&(messages[msg.sender][id].receivers
[receiverToCancel].state==State.finished)){
cancelEvent(receiverToCancel,messages[msg.sender]
[id].receivers[receiverToCancel].hB);
} else if (!messages[msg.sender][id].receivers
[receiverToCancel].isValue){
addReceiver(id,msg.sender,receiverToCancel,
"","",State.cancelled);
cancelEvent(receiverToCancel,stateToString(messages
[msg.sender][id].receivers[receiverToCancel].
state));
} else {
cancelEvent(receiverToCancel,stateToString(messages
[msg.sender][id].receivers[receiverToCancel].state));
}}} else {createMessage(id,msg.sender);
for(uint j= 0; j<cancelledReceivers.length;j++){
address receiverToAdd = cancelledReceivers[j];
addReceiver(id,msg.sender,receiverToAdd,"","",
State.cancelled);
cancelEvent(receiverToAdd,stateToString(messages[msg.sender]
[id].receivers[receiverToAdd].state));}}}

function finish(uint id,address sender,address receiver,
string _hB,string _keyB){
if (msg.sender==ttp){
if (messages[sender][id].isValue){
if((messages[sender][id].receivers[receiver].isValue)
&&(messages[sender][id].receivers[receiver].
state==State.cancelled)){
finishEvent(stateToString(messages[sender][id].
receivers[receiver].state));
}else if(!messages[sender][id].receivers[receiver].isValue){
addReceiver(id,sender,receiver,_hB,_keyB,State.finished);
finishEvent(stateToString(messages[sender][id].
receivers[receiver].state));
}else{finishEvent(stateToString(messages[sender]
[id].receivers[receiver].state));
}}else{createMessage(id,sender);
addReceiver(id,sender,receiver,_hB,_keyB,State.finished);
finishEvent(stateToString(messages[sender][id].
receivers[receiver].state));}}}
```

```
function createMessage(uint id, address sender) private {
messages[sender][id].id=id;
messages[sender][id].sender=sender;
messages[sender][id].isValue=true;}

function addReceiver(uint id, address sender, address
receiver, string _hB, string _keyB, State _state) private{
messages[sender][id].receivers[receiver].receiver=receiver;
messages[sender][id].receivers[receiver].hB=_hB;
messages[sender][id].receivers[receiver].keyB=_keyB;
messages[sender][id].receivers[receiver].state=_state;
messages[sender][id].receivers[receiver].isValue=true;}
...
}
```

Smart Contract 3. Multiparty Protocol

This smart contract allows the association of multiple receivers to a message with a new data structure that includes the required parameters and the state of the exchange. This way, the state is not related only to the message, now it depends of each receiver. The set of receivers will be included in the *struct* using a *map*. This new organization requires important modifications in functions *Cancel* and *Finish*. *Cancel* has an array of receivers as parameter, B". The function cancels the exchange for each receiver or, if it is the case, return the corresponding h_B. *Finish* has to check if B_i is a member of $B" - cancelled$. If it is the case, the TTP will be notified. Otherwise, B_i is included in $B" - finished$ and the NRR proof h_{Bi} and the ciphered key k are stored.

V. Analysis of Properties

The main properties achieved by the multiparty fair confidential certified notification protocol are analysed here.

- **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 intervention of the TTP can lead to a situation where one of the parties possesses contradictory evidence. A malicious A can have the non-repudiation proof received directly from a receiver of B and also the cancellation proof generated by the smart contract after a cancellation request from A. For this reason, the fairness will be weak and the generated proofs are non transferable. However, 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.
- **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 (receivers 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 the members of B have the element received in the third step or the state of the smart contract is *Finished*. A member of B cannot deny having received the notification since A has the elements sent by the members of B in the third 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 functions of the smart contract are executed, then the TTP will process the received elements and will make a transaction including the element that will allow the members of B*"-finished* 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 and the confidentiality will be preserved.

VI. Performance Analysis

One of the key aspects of the development of blockchain-based applications is the performance analysis, due to the fact that the code is not executed locally but in a distributed P2P network. This means that the execution computational cost is translated into an economic cost and a delay.

We have evaluated the performance of a multiparty notification comparing the multiple use of a two party protocol (Smart Contract 1), the use of a reusable two-party protocol (Smart Contract 2) and the use of a multiparty protocol (Smart Contract 3), obtaining some conclusions.

TABLE III
GAS PER EXECUTION

	Two-Party	Reusable	Multiparty
Deployment	800.433	1.010.296	2.150.409
Finish (Cancelled)	26.388	29.821	31.021
Finish	88.387	135.620	179.386
Cancel	44.698	102.011	146.772
Cancel (Finished)	24.772	26.670	86.864
Cancel (10 Users)	-	-	712.707
Cancel (50 Users)	-	-	3.198.072

TABLE IV
USD COSTS

	Two-Party 1Gwei–20Gwei	Reusable 1Gwei–20Gwei	Multiparty 1Gwei–20Gwei
Deployment	0.37 $ - 7.83 $	0.46 $ - 9.89 $	1.00 $ - 21.05 $
Finish (Cancelled)	0.01 $ - 0.26 $	0.01 $ - 0.29 $	0.01 $ - 0.30 $
Finish	0.04 $ - 0.86 $	0.06 $ - 1.32 $	0.08 $ - 1.75 $
Cancel	0.02 $ - 0.44 $	0.05 $ - 0.99 $	0.07 $ - 1.44 $
Cancel (Finished)	0.01 $ - 0.24 $	0.01 $ - 0.26 $	0.04 $ - 0.85 $
Cancel (10 Users)	-	-	0.33 $ - 6.97 $
Cancel (50 Users)	-	-	1.48 $ - 31.30 $

The measures shown in Table III correspond to the fixed costs of gas of each one of the methods that contain the contracts. The most expensive functions are the deployment of contracts. Another aspect to highlight is that the same method can have different costs, despite being executed with the same parameters. This is because when using if-else structures the amount of code that is executed is not always the same. Therefore, we find two different situations. As we can see, the invocation of the *Finish* method if the exchange is *Cancelled* is not the same as if it is not *Cancelled*. This is because in case of being *Cancelled*, it is not necessary to carry out any action except the invocation of the corresponding event. On the other hand, if it is not *Cancelled*, it is necessary to store h_B and the encrypted k key. Additionally, it is also notable how the reusable (Smart Contract 2) and the multiparty (Smart Contract 3) have a higher completion cost. This is logical because, in addition to the storage of the parameters, it is necessary to create the message instance within the contract.

As expected, the cost of the *Cancel* also depends on the state in which the transaction is located. As a result, it will be less expensive to make the cancellation when the exchange is already completed. Another notable factor in multiparty exchange is that the cost is influenced by the number of recipients that are being canceled. In this way, the higher the number, the higher the total cost.

Once the gas costs are established by method, the comparison between protocols can be carried out. As already detailed, the contract without identifiers, (Smart Contract 1) is the one with the lowest cost of gas per deployment. However, it is the least efficient because the TTP must deploy a contract for each one of the exchanges, even if no disputes occur and the contract is not used later. It can be seen how the cost of deployment in the multiparty contract, (Smart Contract 3) is approximately double than the required for the reusable contract, (Smart Contract 2). However, it is important to consider that this disbursement will be made only once and will allow the management of multiple exchanges. Therefore, it should not be considered as a crucial factor in the comparison, but should be taken into account.

In the same way, the *Finish* subprotocol does not suppose a notable difference between the protocols. As can be seen, the gas costs of all the contracts are very similar since the finalizations, even in the case of the multiparty contract, are made individually. The differential factor between the contracts, therefore, will be the *Cancel* subprotocol. The multiparty contract allows cancellation with multiple recipients simultaneously and the cancellation cost per user is approximately 65,000 gas units. On the other hand, in the case of the reusable contract, each individual cancellation has an amount of 102,011 units. This means a difference of approximately 35,000 units. Therefore, for a large number of users, this amount can be considered remarkable.

Taking this into account, it can be concluded that for individual exchanges the use of the reusable contract (Smart Contract 2) is more efficient, since in addition to presenting a lower deployment cost, it also has lower cancellation costs for a single recipient. On the other hand, for exchanges between a sender and multiple recipients it is advisable to use the multiparty contract (Smart Contract 3). This is due to the fact that, despite having a higher deployment cost, it is important to consider that the cancellations are carried out only by

the sender and it this party who has to face the payment. Therefore, a higher initial investment can reduce the cost over end users in the long run.

Table IV shows the orientative economical cost in $ of each protocol[2]. It includes a value related to a gas price of 1 Gwei and another one for 20 Gwei. The gas price determines how likely the transaction will be incorporated in a mined block. Thus, if the users are not in a hurry, they could leave the price of gas at 1 Gwei and the confirmation of the transaction should not take more than 10 minutes (depending on the network congestion). But if they would like a quick confirmation (20 seconds in ideal conditions), they have to increase the price of gas. Thus, the use of a blockchain-based solution can be quick and non-expensive regarding previous solutions with a strong involvement of a TTP [13], [22].

VII. Conclusions

Previous solutions for multiparty 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 recipients have 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 an alternative for sending certified notifications on a blockchain-based fairness. Now trust is achieved in a smart and auto-regulated environment. The presented multiparty fair exchange protocol allows users to send confidential notifications to a set of receivers 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 multiparty fair exchange and achieves 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 one of the parties but also has to check the blockchain. Nevertheless, as a future work, we are planning to make a deeper analysis of the security properties of the proposal. The paper includes three versions of the smart contract associated with the fair exchange protocol, and the performance of each smart contract have been analysed (both in terms of gas and money), defining the ideal solution for each scenario.

Acknowledgements

This work has been partially financed by Spanish government under project AccessTur TIN2014-54945-R.

[2]Measures have been calculated using the exchange rate of June 23, 2018.

References

[1] Asokan, N., Shoup, V. and Waidner, M.: "Asynchronous Protocols for Optimistic Fair Exchange"; Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 86-99, 1998.

[2] Barber, S., Boyen, X. Shi, E. and Uzun, E., "Bitter to Better — How to Make Bitcoin a Better Currency". Financial Cryptography and Data Security, Springer Berlin Heidelberg, p.399-414, 2012.

[3] Chaudhry, S., Farash, M., Naqvi, H. and Sher, M. "A secure and efficient authenticated encryption for electronic payment systems using elliptic curve cryptography". Journal of Electronic Commerce Research, Vol. 16, Num. 1, p. 113-139, 2016.

[4] Delgado-Segura, S., Perez-Sola, C., Navarro-Arribas G. and Herrera-Joancomarti, J. "A fair protocol for data trading based on Bitcoin transactions", Future Generation Computer Systems, 2017.

[5] Ethan H., Baldimtsi, F., Alshenibr, L., Scafuro, A. and Goldberg, S.: "TumbleBit: An Untrusted Tumbler for Bitcoin-Compatible Anonymous Payments". Network and Distributed System Security Symposium, 2017.

[6] Ferrer-Gomila, J., Payeras-Capellà, M. and Huguet i Rotger, L.: "An Efficient Protocol for Certified Electronic Mail". ISW 2000: 237-248, 2000.

[7] Ferrer-Gomila, J., Payeras-Capellà, M. and Huguet i Rotger. L.: "A Realistic Protocol for Multi-party Certified Electronic Mail". ISC 2002: 210-219, 2002.

[8] Ferrer-Gomilla, J., Onieva, J. Payeras-Capellà, M. and Lopez. J: "Certified electronic mail: Properties revisited", Computers & Security, vol. 29, n. 2, p.167-179, 2010.

[9] Heilman, E., Baldimtsi, F. and Goldberg, S. "Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions".Financial Cryptography and Data Security, Springer Berlin Heidelberg. p. 43-60, 2016.

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

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

[12] Kremer, S., Markowitch, O.: "Selective receipt in certified e-mail". LNCS 2247. Advances in cryptology: indocrypt. Springer-Verlag, p. 136-48, 2001.

[13] Kremer, S. and Markowitch, O.: "A multi-party non-repudiation protocol". In Proceedings of SEC 2000: 15th International Conference on Information Security. IFIP World Computer Congress, p.271-280, 2000.

[14] Liu, J., Li, W., Karame, G. and Asokan, N.: "Towards Fairness of Cryptocurrency Payments", IEEE Security and Privacy, 2017.

[15] Mut-Puigserver, M., Ferrer-Gomila, J. and Huguet-Rotger, L.: "Certified e-Mail Protocol with Verifiable Third Party". Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service. p. 548-551, 2005.

[16] Mut-Puigserver, M., Payeras-Capellà, M., Cabot-Nadal, M.: "Blockchain-Based Fair Certified Notifications: ESORICS 2018, Data Privacy Management, Cryptocurrencies and Blockchain Technology, DOI: 10.1007/978-3-030-00305-0_2, 2018.

[17] Onieva, J., Zhou, J. and Lopez J.: "Enhancing certified email service for timeliness and multicast". International network conference, pp. 327-35, 2004.

[18] Onieva, J., Zhou, J., and Lopez, J.: "Multi-Party Non-repudiation: A survey", ACM Computer Surveys, vol. 41, pp. 5, 2008.

[19] Payeras-Capellà, M., Mut-Puigserver, M., Ferrer-Gomila, J. and Huguet-Rotger, L.: "No Author Based Selective Receipt in an Efficient Certified E-mail Protocol". Prallel and Distributed Processing, PDP'09: 387-392, 2009.

[20] Shao, Z. "Fair exchange protocol of Schnorr signatures with semi-trusted adjudicator"". Computers & Electrical Engineering, 2010.

[21] Zhou, J., Deng, R. and Bao, F.: "Some Remarks on a Fair Exchange Protocol"; International Workshop on Practice and Theory in Public Key Cryptosystems, LNCS 1751, Springer Verlag, p. 46-57, 2000.

[22] Zhou, J., Onieva, J. A., and Lopez, J.: "Optimised multi-party certified email protocols". Information Management & Computer Security Journal 13, 5, 350-366 2005.