# Replay Attack in a Fair Exchange Protocol*

Macià Mut-Puigserver, Magdalena Payeras-Capellà, Josep Lluís Ferrer-Gomila, and Llorenç Huguet-Rotger

Universitat de les Illes Balears, Carretera de Valldemossa, km 7,5. 07120 Palma de Mallorca, Spain
{macia.mut,mpayeras,jlferrer,l.huguet}@uib.es

**Abstract.** A fair multi-party exchange protocol provides equal treatment to all users, in such a way that at the end of the execution of the exchange, all parties have the element that wished to obtain, or none of them has obtained any valid item. In this paper, we analyse a well-known multi-party fair exchange protocol and, in spite of the formal proof of its correctness given in [11], we demonstrate that the protocol has a flaw. The weakness provoked by this flaw made possible a replay attack that breaks the fairness of the exchange. We will see as a group of colluding participants in the exchange can get the item from an honest participant and this participant will get nothing. In addition to that, we propose a new protocol to solve the problem of the potential replay attack which preserves the property of *semi-trusted neutral party*. The property was introduced in the original protocol so as to improve the user confidence in the trusted third party (TTP). Our solution not only preserves this property but also introduces the property of *verifiable TTP*. The property guaranties evidences from each TTP operation to the users. The evidences can be used to get compensation and correct any wrong situation caused by an incorrect operation of the TTP; for instance, in case of a passive conspiracy of the TTP.

## 1 Introduction

Some electronic services require an exchange of elements between two or more users. A fair exchange of values always provides an equal treatment to all users, and, at the end of the execution of the exchange, all parties have the element that wished to obtain, or the exchange has not been solved successfully (in this case, nobody has its expected element). Among the electronic applications that require a fair exchange of information we can find electronic contract signing, certified electronic mail and electronic purchase (payment in exchange for a receipt or a digital product).

In a fair exchange protocol several entities wants to exchange their goods in such a way that, at the end of the protocol, any honest participant has received

* This work was supported by ARES: Advanced Research on Information Security and Privacy (Consolider-Ingenio-2010 Program, 2007-2012), Seguridad en la contratación electrónica basada en servicios web (CICYT TSI2007 62986) and PROGECIB-16A (CAIB).

S.M. Bellovin et al. (Eds.): ACNS 2008, LNCS 5037, pp. 174–187, 2008.

all expected items or none of them has obtained any valid item. Fair exchange protocols often use trusted third parties (TTPs) helping users to successfully realize the exchange. So, TTPs play an important role in these protocols and their reliability and security is a problem that needs to be addressed, because the security of the exchange can be broken if the TTP doesn't work properly. In this paper, we will break the fairness of the exchange of an existing protocol using a replay attack against the TTP. As a result, the protocol will finish with an entity, who has given away his own item, but without getting the other expected item.

We can distinguish between single-unit and multi-unit exchange protocols. In the case of multi-unit exchange protocols different topologies are possible (star, ring, matrix, graph). Protocols like [2, 5] suppose that each party exchanges an item against another and the exchange's topology is a ring. Franklin and Tsudik in [5] propose a new multi-party fair exchange protocol with a ring topology, which was improved in [7] and a final version of the protocol with a proof of its correctness was given in [11]. Here, we analyse the protocol proposed by Mukhamedov et al. in [11] and we will demonstrate that, in spite of the improvements suggested and the formal proof of correctness, the security of the exchange can be broken using a replay attack.

Our solution to the replay attack not only preserves a TTP with the quality of *semi-trusted neutral party* [5] but also she is *verifiable* [12]. This key feature removes a weakness of the protocol (e.g.; it can defend users against a passive conspiracy of the TTP) and increases the user's confidence in the TTP. Our new scheme will provide users with evidences of the TTP operations. In case of any misbehaviour of the TTP, participants can make use of the evidences to correct the unfair situation.

The paper is organized as follows: section 2 presents some security notions and the multi-party fair exchange protocol, section 3 describes what we can consider a reply attack, sections 4 and 5 describes the replay attack and its solution. Section 6 shows that the TTP involved in the new protocol is verifiable. Section 7 is devoted to the conclusions of this work.

## 2 Multi-party Fair Exchange Protocol

### 2.1 Syntax

To give a description of the protocols we use the notation *number*.{*event*}: {*description*} to describe the individual steps of these protocols, where *number* is the step number of the protocol, {*event*} can be the sending of a message from user $X$ to $Y$ (designated by $X \rightarrow Y$) or an *ftp get* operation (designated by $X \leftrightarrow Y$) and it can also be some local computation of a participant. The notation of the encryption operations will be:

- $PU_A(m)$ is the result of applying an asymmetric encryption algorithm to the plaintext $m$ under Alice's public key;

- $PR_A(m)$ denotes the digital signature of Alice over message $m$ using her private key;
- $E_K(m)$ is the symmetric encryption of message $m$ under key $K$.

The encryption schemes $E_K(), PU_A()$ and $PR_A()$ are not homomorphic.

### 2.2 Security Notions

In this section we will give a general definition of the security of an exchange protocol. As suggested in [10], we say that an exchange protocol is *secure* when it respect these three mandatory properties:

1. *Viability*: independently of the communication channels quality, there exists an execution of the protocol, where the exchange succeeds.
2. *Fairness*: the communication channels quality being fixed, at the end of the exchange protocol run, either all involved parties obtain their expected items or none (even a part) of the information to be exchanged with respect to the missing items is received.
3. *Timeliness*: the communication channels quality being fixed, the parties always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness.

So, a secure exchange protocol must be fair relating to the exchanged items and it also must be fair relating to the ability to determine the progress of the protocol (called timeliness). Moreover, a secure exchange can respect some optional properties, like non-repudiation, abuse-freeness and verifiability of the TTP [10, 12, 14]. These additional properties allow the evolution of fair-non-repudiation protocols in the critical security points, such as the trusted third party's involvement in fair exchange schemes.

### 2.3 Protocol Description

We will describe the multi-party fair exchange protocol proposed in [11] which will be the object of our replay attack. To do that, we need to know how the privacy and the authentication properties are achieved in this protocol, because the authors in [11], for the sake of simplicity, didn't give explicit specifications about that. For this reason we have chosen the description of the same protocol made in [7] as a way of achieving privacy and authenticity. Of course, we have added the two fixes proposed in [11] (the solution to the possible collision in the label space and the fix to the arithmetic attack that they describe).

Different topologies are possible in the case of multi-party fair exchange protocols. The exchange topology in the protocol proposed in [11] is a ring. In this case, each participant $P_i (i \in [1, n])$ desires an item (or a set of items) from the participant $P_{i-1}$ and offers an item to the participant $P_{i+1}$. The communication channels between participants are unreliable and those used between each participant and the TTP are resilient (a communication channel is resilient if a message inserted into such a channel will eventually be delivered [14]).

The protocol relies on a homomorphic one-way function $f$ (*i.e.*; $f(x_1 \cdot x_2) = f(x_1) \cdot f(x_2)$) and a function with $n$ arguments $F_n$ such that:

$$F_n(x_1, f(x_2), \ldots, f(x_n)) = f(x_1 \cdot x_2 \cdot \ldots \cdot x_n)$$

The proposed function $f$ is $f(x) = x^2 (mod\, N)$ and $F_n(x_1, x_2, \ldots, x_n) = x_1^2 \cdot x_2 \cdot \ldots \cdot x_n$ where $N$ is an RSA modulus. To avoid notational clutter, we suppose that all operations on subscripts are performed modulo $N$. As we have said, the exchange is cyclic, so, an entity $P_i, i \in [1, n]$, sends his secret information $m_i$ to $P_{i+1}$ in exchange of $P_{i-1}$'s secret information $m_{i-1}$. At the end of setup phase the authors suppose that:

- Each participant knows the identity of the remaining participants in the exchange;
- All participants agree on the TTP and the functions $f$ and $F_n$;
- Descriptions of the items to be exchanged $f(m_i)$ are public.

A label $l$ has been introduced to identify a protocol's run. To avoid possible collision in the label space, Mukhamedov et al. explain that the TTP will generate and distribute the value of $l$ to the participants of the exchange at the setup phase (timestamps or nonces may suffice for this purpose together with the TTP's name). The protocol is as follows:

1. $P_i \rightarrow P_{i+1} : PR_{Pi}(l, PU_{Pi+1}(R_i))$
2. $P_i \rightarrow TTP : PR_{Pi}(l, A_i, PU_T(C_i), f(R_i), f(R_{i-1}), f(m_{i-1}))$,
   where $A_i = F_n(m_i, f(m_1) \ldots f(m_{i-1}), f(m_{i+1}) \ldots f(m_n))$ and $C_i = m_i \cdot R_i^{-1}$
3. $TTP \rightarrow P_i : PR_T(l, \mathbf{C})$,
   where $\mathbf{C} = \{C_i \mid 1 \leq i \leq n\}$

Each participant $P_i$ chooses a random value $R_i$, then she encrypts it and sends it to $P_{i+1}$. Then, each $P_i$ computes $C_i = m_i \cdot R_i^{-1}$ and $A_i$ and she sends this information to the TTP along with $f(R_i), f(R_{i-1})$ and $f(m_{i-1})$. The TTP waits until it has received the $n$ messages from the participants and then she performs the following checks:

*a.* The $f(R_i)$ sent by $P_i$ is equal to the $f(R_i)$ sent by $P_{i+1}$;
*b.* All received $A_i$ are equal. If so, the TTP computes $\zeta = C_1 \cdot \ldots \cdot C_n$ and $F_{n+1}(\zeta, f(R_1), \ldots, f(R_n))$. This second computation has to be equal to any $A_i$. These operations are made by the TTP to ensure consistency between $m_i$ and $f(m_i)$;
*c.* The TTP must be sure that $C_i$ contains $R_i^{-1}$. So, the TTP checks that each $C_i \in \mathbf{C}$ verifies: $f(C_i \cdot R_i) = f(m_i), \forall i \in [1, n]$.

If checks succeed then the TTP will send $\mathbf{C}$ to the participants. This enables each $P_i$ to compute $m_{i-1} = C_{i-1} \cdot R_{i-1}$.

## 3 Replay Attacks

Replay attacks have been discussed for quite some time in the literature (e.g.; [1, 6, 13]). A reply attack is a kind of active attack where a user (the attacker) records a communication session or part of it and then, later, replays the entire session, or a portion of the recorded information so as to take advantage of it.

The attacker can be a single user involved in the exchange (e.g.; a sender or a receiver of a certified electronic mail, a payer or a payee in an electronic purchase, a contract signer...) or a confabulation of users.

Attackers want to have a benefit replaying the exchange (or part of it). This benefit, in case of fair exchange, can be obtaining the token they wanted to receive through the exchange. In order to get the element, the attacker usually contacts with the TTP and sends him some information included in the recorded exchange. As a result of this attack, the attackers gain the element they desired to obtain and some other honest participant will not receive the expected element.

## 4 Replay Attack to the Multi-party Fair Exchange Protocol

A replay attack against the TTP could be possible in the protocol defined in §2.3. In this attack, a group of dishonest participants make the TTP believe that the exchange is stopped. Then, they start a new protocol's run without an honest participant of the previous exchange and they use the TTP to get the item of this participant from the first exchange. We'll attempt to illustrate the attack by giving this example:

1. Suppose we have three participants involved in a cyclic exchange using the protocol specified in [11]: $P_i$ sends her information $m_i$ to $P_{i+1}$ in exchange of $P_{i-1}$'s secret information $m_{i-1}(\forall i = 1..3)$; to close the ring $P_3$ sends her information to $P_1$. We assume that $P_1$ collude with $P_2$ against $P_3$.
2. The second step of the protocol will not be performed neither by $P_1$ nor by $P_2$. However, they capture the message sent by $P_3$ to the TTP:

$$PR_{P3}(l, A_3, PU_T(C_3), f(R_3), f(R_2), f(m_2))$$

   The TTP will not receive any other message of this protocol's run, so she eventually stops this protocol's instance.
3. Afterwards $P_1, P_2$ and $P_4$ (a new dishonest participant colluded with $P_1$ and $P_2$) start a new protocol's run. The target of this new exchange is getting the item $m_3$ of $P_3$ from the previous exchange. In this case, $P_1$ and $P_2$ are supposed to exchange new items (the description of the items are $f(\widetilde{m}_1)$ and $f(\widetilde{m}_2)$ respectively), but these participants deceive the TTP, they say that the description of the item exchanged by $P_4$ is $f(m_3)$. Thus, after the exchange of the random numbers at the first step of the protocol, the second step will be as follows:

$$P_1 \rightarrow TTP : PR_{P1}(\widetilde{l}, \widetilde{A}_1, PU_T(\widetilde{C}_1), f(\widetilde{R}_1), f(R_3), f(m_3))$$
$$P_2 \rightarrow TTP : PR_{P2}(\widetilde{l}, \widetilde{A}_2, PU_T(\widetilde{C}_2), f(\widetilde{R}_2), f(\widetilde{R}_1), f(\widetilde{m}_1))$$
$$P_4 \rightarrow TTP : PR_{P4}(\widetilde{l}, \widetilde{A}_1, PU_T(C_3), f(R_3), f(\widetilde{R}_2), f(\widetilde{m}_2))$$

The information $PU_T(C_3), f(R_3)$ and $f(m_3)$, used in the new messages, is taken from the message captured in the previous protocol's run. The value of $\widetilde{A}_{i=1,2,4}$ must be the same, so $P_4$ copies this value from $P_1$ because it is impossible to be computed by $P_4$ without knowing $m_3$.

4. Finally, the TTP checks:
   (a) The $f(R_i)$ sent by $P_i$ is equal to the $f(R_i)$ sent by $P_{i+1}$;
   (b) All received $A_i$ are equal. The TTP computes $\zeta = \widetilde{C}_1 \cdot \widetilde{C}_2 \cdot C_3$ and checks that $F_4(\zeta, f(\widetilde{R}_1), f(\widetilde{R}_2), f(R_3))$ is equal to any $A_i$;
   (c) The TTP also has to check that each $C_i \in \mathbf{C}$ verifies: $f(C_i \cdot R_i) = f(m_i), \forall i \in [1, n]$; where $\mathbf{C} = \{C_i \mid 1 \leq i \leq n\}$.
5. Participants will receive $\mathbf{C}$ from the TTP. Participant $P_1$ has received $R_3$ in the first step of the previous protocol's run. Thus, $P_1$ can compute the message $m_3 = C_3 \cdot R_3$.

Following these five steps, participant $P_1$ has got $m_3$ from $P_3$ and $P_3$ has not received the message expected. So, the fairness of the exchange has been broken.

## 5 Solving the Replay Attack

### 5.1 Fixing the Protocol

We will modify the original protocol so as to solve the problem of the replay attack. As a result, we will get a new protocol resistant to this kind of attacks. In the new scheme, before starting the exchange phase of the protocol, there is a setup phase. At the end of this phase:

*a.* Participants in the exchange have agreed on the identity of the TTP and on the functions $f$ and $F_n$.
*b.* Each participant knows the identity of the remaining participants in the exchange. The TTP knows these participants and publishes a list with their identities ($\rho = PR_T(l, P_1, P_2, \ldots, P_n)$ is the set of all participants with the label of the exchange $l$, which identifies a protocol's run). The order of appearance in this list determines the way in which participants are arranged in the ring topology of the exchange.
*c.* Participants also send the description of items that will be exchanged to the TTP ($f(m_i), \forall i \in [1, n]$). The TTP publishes a list with these descriptions: $\delta = PR_T(l, f(m_1), f(m_2), \ldots, f(m_n))$.

Thus, at the end of the setup phase, participants can download from a public directory managed by the TTP the following items:

0. $P_i \leftrightarrow TTP : \rho, \delta$

Then, each user $P_i$ verifies the correctness of $(\rho, \delta)$ and begins the exchange phase by choosing a random value $R_i$ and sending it to $P_{i+1}$ as follows:

1. $P_i \rightarrow P_{i+1} : PR_{Pi}(PU_{Pi+1}(l, R_i))$
2. $P_i \rightarrow TTP : PR_{Pi}(l, A_i, E_{Ki}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1}))$, where:

$$C_i = m_i \cdot R_i^{-1}, and$$
$$A_i = F_n(m_i, f(m_1), \ldots f(m_{i-1}), f(m_{i+1}), \ldots, f(m_n))$$

Note that label $l$ is encrypted along with the random number $R_i$, the key $K_i$ and the message $C_i$. Thus, nobody can use $R_i$ or $C_i$ outside the scope of this exchange. Inside the messages sent by participants to the TTP, $f(m_{i-1})$ was introduced in [11] because the TTP needs to know agent-message correspondence. But now, this information is known by the TTP during the setup phase (in fact, TTP publishes $\rho$ and $\delta$).

In order to know the status of the exchange and to verify the task of the TTP, the TTP publishes, every period of time $\Delta t$ (this period is a public parameter of the TTP), an authenticated list $(\lambda)$ of received messages from the participants by the TTP. The items issued by the TTP are published in a public directory managed by this entity. When all messages are received, the TTP verifies that these messages have the correct format with the appropriate label, items, signatures and encryptions. Obviously, that is very important to detect any kind of attack, but, here, we want to emphasize the role of the correctness of these verifications in messages $PR_{Pi}(l, A_i, E_{Ki}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1}))$ in order to prevent the replay attack: any participant cannot separate $C_i$ from $l$, which was the basis of the replay attack of §4.

Next, the TTP performs the verifications specified in §2.3 at the step three of the protocol and, if all checks succeed, she publishes $\mathbf{C}$ and $\mathbf{K}$ as follows:

3. $P_i \leftrightarrow TTP : PR_T(l, \mathbf{C} = \{C_i | 1 \leq i \leq n\}, \mathbf{K} = \{K_i | 1 \leq i \leq n\})$

Finally, each participant $P_i$ computes $m_{i-1} = C_{i-1} \cdot R_{i-1}$ and obtains the expected item. However, an execution of the protocol can be aborted by any participant if the TTP has not received the corresponding messages (step 2) from all participants and, therefore, she has not published $\mathbf{C}$ (step 3). To cancel an exchange, any participant has to verify that the TTP has not collected all messages (i.e. the last published list $\lambda$ doesn't contain all messages and $\mathbf{C}$ is not published), then she can perform this alternative step three:

3. $P_i \rightarrow TTP : PR_{Pi}(l, ''cancel'')$
4. $P_i \leftrightarrow TTP : PR_T(l, \lambda, ''cancelled'')$

When the TTP receives a message to cancel an exchange from a participant, she checks whether the last published $\lambda$ has all messages from the participants. If the check fails, the TTP makes public the message $PR_T(l, \lambda, ''cancelled'')$ instead of publishing the next list of received messages $\lambda$. Obviously, the list

$\lambda$ in message $PR_T(l, \lambda, ''cancelled'')$ must not contain all messages of the step 2 but it has to contain the request to cancel the exchange: $PR_{Pi}(l, ''cancel'')$. Lastly, participants can get the message $PR_T(l, \lambda, ''cancelled'')$, which is an evidence that prove that the exchange identified by label $l$ has been aborted.

### 5.2 Security of the Protocol

With respect to the three mandatory properties of a secure fair exchange we can state that this new scheme is secure because:

1. Obviously the three step of the new protocol has an execution where the exchange succeeds. So, the protocol is *viable*.
2. According to the definitions in §2.2, the proposed protocol is fair. A protocol's run always ends with a fair situation:
   (a) If everyone is honest, then each $P_i$ can compute: $m_{i-1} = C_{i-1} \cdot R_{i-1}$, at the end of the protocol.
   (b) If all parties are honest but the TTP is not, all $R_i$ values are predistributed securely and the TTP cannot obtain any $R_i$. Moreover, if the TTP misbehaves, participants will have evidences to prove that and to correct the unfair situation (see §6: verifiable third party).
   (c) If a protocol run is aborted, a malicious participant cannot take advantage of that. Any participant will not be able to use the information of this protocol's run outside the scope of the exchange (the information items used in the protocol are linked to the exchange and the encryptions algorithms are not homomorphic, so it is not possible to use any piece of information from a particular exchange in another exchange without being detected). Thus, in this case, the malicious participant cannot learn the secret information from another participant.
3. Participants has the ability to determine the progress of the protocol and they can arrive, in finite amount of time, at a point where the protocol stops preserving fairness (here, we suppose that the TTP will reply to any message from a participant in a finite amount of time). For example, if the TTP doesn't publish **C** and **K** because she has not received all messages from participants, then any participant can cancel the protocol preserving the security of the exchange.

## 6 Verifiable Third Party

Besides the compulsory properties, some other remarkable properties can be present in a fair exchange. In order to improve the user confidence in the TTP, the original protocol was presented in [5] with an important property: the TTP is a *semi-trusted neutral party*. Numerous papers [3, 4, 5, 8, 9] express the concern about that, because it is necessary the user confidence in the critical security points in order to spread the use of new electronic procedures. From this point of view, we need to design new protocols where the user confidence on what

is happing when he invokes the TTP is enhanced. Otherwise, it could be very difficult to find designated full-time neutral parties to rely on in order to use any protocol.

A *semi-trusted neutral party*, as it is explained in [5], can be selected on a case-by-case basis and asked to aid in the execution of a fair multi-party exchange. However, while a TTP with this property is trusted to ensure the fairness of the exchange, it is not trusted with the actual items involved in the exchange. This means that a malicious semi-trusted neutral party must be unable to cheat as long as the other parties remain honest (i.e. a malicious TTP cannot know and reveal the content of the exchanged items).

Our solution, the proposed protocol in §5.1, is designed to achieve another property in order to increase the user confidence on the TTP operations. The TTP involved in this protocol is *verifiable* according to the concepts defined in [12]. A *verifiable TTP* aims to reduce the amount of trust that users have to place in TTPs, because *verifiability* is a security capability, that is to say, it is a mechanism to protect users against security threats. Now, we are going to give a definition of a verifiable TTP according to the concepts given in [12]:

**Definition 1.** *A security service is verifiable if the user, who sends a request to a TTP, receives a non-repudiation evidence of each operation carried out by the TTP to provide the service.*

**Definition 2.** *The verifiability of a security service is on-line if the user, after checking the received evidences, can immediately know whether the TTP misbehaved. In case of problems, the user can start a dispute to correct the situation.*

**Definition 3.** *A TTP is verifiable if the security services it provides are verifiable and the verifiability of the services is on-line.*

Thus, to be verifiable, a TTP has to issue evidences about its operation. For instance, if the fairness of the exchange is lost due to an incorrect TTP operation, then users will have evidences to start a dispute in order to restore the fairness.

To specify the protocol of the above section, we have followed the guidelines given in [12] so as to have a verifiable TTP operation. As a result, the TTP not only is a *semi-trusted neutral party* (she doesn't know any $R_i$ to decrypt messages) but also is *verifiable* (any wrong operation of the TTP can immediately be detected and participants have evidences of this wrong operation. The proof can be used to achieve fairness in an external dispute resolution system).

### 6.1 Proof of the Verifiability

In order to show that the TTP is verifiable, we have to prove that the service it provides it's also verifiable. Previous to that, at the step 0 of the protocol the TTP issues lists $\rho$ and $\delta$. The lists are authentic (they have the TTP's signature). Anyone can check the correctness of the participants in the ring topology of the exchange and also the correctness of the exchanged item descriptions. Any participant, who does not agree with these lists, must not start with the protocol. She has to inform the rest of participants and the TTP about that.

**Proposition 1.** *The TTP involved in the protocol specified in §5.1 is verifiable.*

*Claim 1.* The activity performed by the TTP at the second step of the protocol is on-line verifiable.

*Proof.* The TTP receives the messages from each participant. Every period of time $\Delta t$, the TTP publishes the received messages during the interval in the list $\lambda$. Anyone can get $\lambda$ and, in case of communication problems, any participant can know if the TTP has received his message or not. We have to remember that the channel is resilient and there is no deadline in the protocol, thus any message sent to the TTP will eventually be delivered to this entity. $\lambda$ is a non-repudiation of reception evidence of the user's messages issued by the TTP. Thus, anyone can claim that a received valid message (a message made in accordance with the protocol specifications) and its information are not in the final message published by the TTP where there is $\mathbf{C}$. Each user has the following evidences to solve a dispute about this activity: $\rho, \delta, \lambda$, its message $(PR_{Pi}(l, A_i, E_{Ki}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1})))$ and the final message of the TTP $(PR_T(l, \mathbf{C}, \mathbf{K}))$.

*Claim 2.* At the step three the TTP publishes the list of encrypted messages $\mathbf{C}$. According to the protocol specified in §5.1 this is an on-line verifiable activity.

*Proof.* As we have seen in claim 1, any user has enough evidences to prove that all encrypted messages from participants have to be in this list and in the same order that is was specified at the setup phase (as stated in the ring topology of the participants). Moreover, the corresponding decrypted messages have to agree with the description established at the setup phase. Otherwise the TTP shouldn't have published it in the list, however users have the following evidences to claim for that in a dispute resolution system: $\rho, \delta, \lambda, PR_T(l, \mathbf{C}, \mathbf{K})$.

*Claim 3.* The operation of the TTP when publishes a cancellation token of an execution of the protocol is on-line verifiable.

*Proof.* participants of the exchange are able to check if the TTP has received all messages of the step 2 or not (if the TTP publishes any list $\lambda$ with all these messages then she cannot cancel the protocol because users will have evidences of the incorrect operation). The message $PR_T(l, \lambda, ''cancelled'')$ is an evidence that proves that the TTP has cancelled the exchange because she has received a request to cancel the execution of the protocol from a participant.

**Result.** According to claims 1, 2 and 3, the security activities performed by the TTP to provide the service are on-line verifiable. Thus, the TTP involved in the protocol is verifiable: users will get non-repudiation evidences of the TTP operations. These evidences allows users to verify the service provided by the TTP, that it means that they can immediately verify the items published by the TTP to provide the service and, in case of any incorrect action that breaks the fairness of the exchange, users will have evidences to correct the situation.

### 6.2 Defending Against Passive Conspiracies in Case of Verifiable TTP

The protocol designed by Franklin and Tsudik in [5] has a *semi-trusted neutral party* which guarantees the message confidentiality to the users, in the sense that the TTP will not be able to know the content of items exchanged. Thus, we can say that the TTP only has an *operational* role in the protocol. In addition to that, we not only have improved the security of the protocol but also we have introduced the verifiability of the TTP as an additional property of the protocol. A *passive conspiracy* (PC) occurs whenever a dishonest party (or a group thereof) conspires with an honest party without the latter's consent. One of the possible PC scenarios of two-party fair exchange, which the TTP is involved as the attacker, is:

- The TTP could (without any consent) favor $P_1$ over $P_2$ and cause to learn $m_2$ without $P_2$ learning $m_1$.

Franklin and Tsudik in [5] consider two sub-types of this kind of PC for the multi-party fair exchange:

1. The TTP selectively broadcast (i.e., directs its message to some participants but not to others).
2. The TTP intentionally sends corrupt $\mathbf{C}$ (i.e., some $C_i$ values are genuine and some are fake).

In order to prevent an honest participant from becoming an unwilling co-conspirator of the TTP, the authors in [5] propose a modus operandi for an honest participant in an exchange:

- Each participant $P_i$, before searching for its barter secret in $\mathbf{C}$, computes the product of all elements in $\mathbf{C}$:

$$\zeta = \Pi(\mathbf{C}) = C_1 \cdot \cdots \ldots \cdot C_n$$

- Next, $Pi$ checks:

$$A_i = F_{n+1}(\zeta, f(m_1), \ldots, f(m_n))$$

  - If the check fails, then
    - ▷ An honest $P_i$ must halt the protocol and not compute $m_{i-1}$.
  - Otherwise,
    - ▷ $P_i$ computes $m_{i-1}$ and broadcasts $\mathbf{C}$ to all other participants (for the benefit who may not have received it, perhaps because of a misbehaving TTP).

  As is said in [5], this procedure ensures that participant $P_i$ does not learn its barter secret $m_{i-1}$ while some other participant is denied the opportunity to learn its secret.

However, the TTP involved in the protocol specified in §2.3 is not verifiable. So, participants cannot be sure about the cause of the PC attack described at the previous paragraph, because the check $A_i = F_{n+1}(\zeta, f(m_1), \ldots, f(m_n))$ can be correct even if $C_i$'s are untidy in $\mathbf{C}$. Thus, the solution proposed in [5] it is not correct because only participants, who cannot decrypt her expected message from $\mathbf{C}$, know that the TTP has not provided the service as the protocol specifies but they don't have any evidence to prove it (they only have: $PR_{Pi}(l, PU_{Pi+1}(R_i))$, and $PR_T(l, \mathbf{C})$).

Even these participants cannot say who is responsible for the attack. Because, they don't have evidences to prove whether the problem arise when some colluded participants exchanged their $C_i$'s before sending them to the TTP and the TTP doesn't perform all checks previous publishing $\mathbf{C}$ or all participants are honest but the TTP misbehaves when publishes a list $\mathbf{C}$ where some $C_i$'s are fake.

The problem of a dishonest TTP in the proposed PC scenario can have a different solution if the TTP is verifiable. Using the protocol of §5.1, each participant will have the following non-repudiation evidences:

$\triangleright\ PR_{Pi-1}(PU_{Pi}(l, R_i))$,
$\triangleright\ PR_{Pi}(l, A_i, E_{Ki}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1}))$,
$\triangleright\ \lambda, \rho, \delta$ *and*
$\triangleright\ PR_T(l, \mathbf{C}, \mathbf{K})$

This enables participants not only to verify the correctness of the TTP operations but also they can demonstrate that the TTP has misbehaved if a conspiracy of the above proposed scenario occurs:

- $P_i$ can check that the result of applying $f$ to $K_{i+1}$ published in $\mathbf{K}$ is equal to $f(K_{i+1})$ in the $P_{i+1}$'s message that appears in $\lambda$.
- $P_i$ can also verify that the decryption of $E_{Ki+1}(l, C_{i+1})$ with $K_{i+1}$ has the label of the exchange and the same $C_{i+1}$ that is in $\mathbf{C}$.

Thus, the operation of the TTP has been verified by participants and if the TTP publishes any $C_i$ or $K_i$ different from the originals sent by $P_i$, then participants will detect this misbehaviour and they have evidences to prove it to an external arbiter (e.g. a judge).

## 7 Conclusions

Franklin and Tsudik defined a multi-party exchange protocol in [5]. The protocol was improved in [7] and recently was published with a formal proof of correctness using the strand space formalism [11]. In this paper, we analyse the protocol and, in spite of the improvements and the formal proof of its correctness, we have demonstrated that the protocol had a flaw. The weakness provoked by this flaw made possible a replay attack that breaks the fairness of the exchange. In §4, we

have seen as a group of colluding participants in a multi-party fair exchange can get the item from an honest participant and this participant will get nothing. To break the fairness of the exchange, dishonest participants use what we call a replay attack. A replay attack is possible in this protocol because some items of the messages exchanged in an instance of the protocol can be used in a different instance of the protocol. Then, the dishonest participants use the TTP involved in the protocol as an oracle to achieve the secret information from the honest one. The attack is possible because the private channel between participants and TTP protects the information that must be secret but this information is not linked to a particular run of the protocol.

In §5.1 we have repaired the protocol to solve the problem of the replay attack. In addition to that, the fixed protocol has a procedure which allows users to cancel an exchange. With this procedure we have add the property of *timeliness* to the protocol, then in §5.2 we have been able to demonstrate the security of the new exchange scheme. We think that the proposed fair exchange scheme is a good example how a protocol can be executed over an insecure network and provides an additional property of TTP-verifiability, i.e. a misbehaving TTP can be proven (of course the protocol also have the so-called mandatory properties of a secure fair exchange: viability, fairness and timeliness).

The role of third parties in protocols is very important. In fact, the original protocol in [5] was proposed with an important property related to the TTP: the entity is called *semi-trusted neutral party* because she cannot reveal the content of the exchanged items. The property wants to decrease a potential risk of the protocol in case of an incorrect TTP behaviour (malicious or not) and, as a consequence, the protocol expects to reduce the amount of users reluctant to use the new electronic procedures. Following this idea, our solution to replay attack has been designed to involve a verifiable third party.

The introduction of the verifiability property in security protocols aims to spread the use of such protocols. We think that this property build a trusted infrastructure so as to enhance the user confidence on what is happen during every protocol's run. Thus, the verifiability of the third party is a security property, which can be employed to convince reluctant users in using the new electronic procedures. With verifiable TTPs we can show how a potential weakness of some security protocols can be removed, thereby improving the security of the system. Verifiable third party is a property that provides evidences to the users from each TTP operation in an easily and comprehensible way. The evidences can be used to get compensation and correct a wrong situation caused by an incorrect operation of the TTP in an external dispute resolution system. For instance, in this paper, we have seen as this property protects users against a passive conspiracy of the TTP which pretends to break the fairness of the exchange.

In further works we have to consider the way of formally proving the correctness of a protocol, because, here, we have broken the security of a fair exchange protocol with a formal proof of correctness presented in [11] (the given security proof is only related to the fairness property not with other security properties).

## References


1. Aura, T.: Strategies against replay attacks. In: 10th IEEE Computer Society Foundations Workshop (CSFW 1997), pp. 59–68. IEEE Computer Society Press, Los Alamitos (1997)
2. Bao, F., Deng, R.H., Nguyen, K.Q., Varadharanjan, V.: Multi-party Fair Exchange with an Off-line Trusted Neutral Party. In: DEXA 1999 Workshop on Electronic Commerce and Security, Italy (1999)
3. European Commission Information Society DG. Open Information Interchange (OII) service: OII Guide to Trust Services, `http://www.diffuse.org/oii/en/trust.html`
4. European Telecommunications Standard Institute (ETSI): Telecommunications Security; Trusted Third Parties (TTP); Requirements for TTP Services; ETSI Guide EG 2001 057 v1.1.2 (1997-2007)
5. Franklin, M.K., Tsudik, G.: Secure Group Barter: Multi-Party Fair Exchange with semitrusted neutral parties. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 90–102. Springer, Heidelberg (1998)
6. Gong, L., Syverson, P.: Fail-stop protocols: An approach to designing secure protocols. In: 5th International Working Conference on Dependable Computing for Critical Applications, pp. 44–55 (1995)
7. Gonzalez-Deleito, N., Markowitch, O.: Exclusion-freeness in Multi-party Exchange Protocols. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 200–209. Springer, Heidelberg (2002)
8. Internet Policy Institute. Report of the National Workshop on Internet Voting: Issues and Research Agenda (March 2001), `http://www.internetpolicy.org`
9. ITU-T: Recommendation X.842: Information technology – Security techniques – Guidelines on the use and management of trusted third party services (October 2000)
10. Markowitch, O., Gollmann, D., Kremer, S.: On fairness in exchange protocols. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 451–464. Springer, Heidelberg (2003)
11. Mukhamedov, A., Kremer, S., Ritter, E.: Analysis of a Multi-Party Fair Exchange Protocol and Formal Proof of Correctness in the Strand Space model. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 225–269. Springer, Heidelberg (2005)
12. Mut Puigserver, M., Ferrer Gomila, J.L., Huguet i Rotger, L.: Certified e-mail Protocol with Verifiable Third Party. In: IEEE International Conference on e-Technology, e-Commerce and e-Service EEE 2005, Hong Kong, pp. 548–551 (2005)
13. Syverson, P.: A Taxonony of replay attacks. In: 10th IEEE Computer Society Foundations Workshop (CSFW 1997), pp. 187–191. IEEE Computer Society Press, Los Alamitos (1997)
14. Zhou, J., Deng, R.H., Bao, F.: Evolution of Fair Non-repudiation with TTP. In: Pieprzyk, J.P., Safavi-Naini, R., Seberry, J. (eds.) ACISP 1999. LNCS, vol. 1587, pp. 258–269. Springer, Heidelberg (1999)