Strand spaces and fair exchange: More on how to trace attacks and security problems*

Fabio R. Piva[1], **José R. M. Monteiro**[1,2], **Ricardo Dahab**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6176 – 13084-971 – Campinas – SP – Brasil

[2]Centro de Pesquisa e Desenvolvimento para a Segurança das Comunicações – CEPESC/Abin Brasília – DF – Brasil

fabio.piva@students.ic.unicamp.br, {monteiro, rdahab}@ic.unicamp.br

Abstract. *In this work we use our proposed adaptation of the strand spaces method in the analysis of a fair exchange protocol for payment, proposed in [Zuo and Li 2005]. The protocol fails to provide timeliness and fairness to the buyer (Downloader), and four previously unreported attacks are traced regarding those properties. This is a continuation of the work started in [Piva et al. 2006].*

1. Introduction

Fairness presents a difficult problem in Internet transactions, that can be mitigated by the use of *fair exchange protocols*. Such protocols provide communicating parties with assurances regarding the outcome of the exchange: each party receives the item desired if and only if the other party gets his or hers. Exchanges usually happen over insecure channels, and between mutually distrusting parties. Solutions comprise protocols based on a trusted third party (TTP), with varying degrees of involvement. The optimistic protocols, in particular, rely on *online* TTPs, which are limited to resolving conflicts between parties (as opposed to being involved in the entire communication between them), thus reducing the occurrence of bottlenecks.

According to [Asokan 1998], desirable fair exchange properties are: (i) *effectiveness*; (ii) *fairness*; (iii) *timeliness*; (iv) *non-repudiation* of the actions of each party; (v) *verifiability of the TTP*; and (vi) *abuse freeness*. Exceptions may arise either from misbehavior of one of the parties, or from third party interference, or from faults in the communication channel. Optimistic fair exchange protocols usually enforce timeliness through the use of local timeouts and a TTP to resolve disputes. If that TTP misbehaves, but does not collude with any party, it is called a *semi-trusted* third part ($STTP$).

Fair exchange properties are usually established through informal arguments; however these arguments can be made less suceptive to mistakes by using a formal verification thechnique. The original *strand spaces method* [Thayer et al. 1999b] allows one to represent protocols for authentication and key establishment, as well as to prove authentication-related properties. An adaptation proposed in [Thayer et al. 1999a] allows for parallel executions of the analysed protocol, but forbids exchange of terms in a parallel run of the protocol's main body (which is exactly what occurs in fair exchange protocols). In [Guttman and Thayer 2002], authentication tests are introduced; by the use of test components, the authors show how to determine ambiguities in the origin of terms in entity traces.

*First author supported by CNPq

In [Piva et al. 2006], we show how the strand spaces method can be applied on the verification of fair exchange protocols, in a similar way to that introduced in [Guttman and Thayer 2002]. We suggest how fair exchange properties should be translated into strand spaces theorems and give general examples of each property.

In this work we employ the adaptation proposed in [Piva et al. 2006] to show how a supposedly fair payment protocol fails to achieve fairness and timeliness. Specifically, we will use strand spaces to analyse the Zuo and Li's p2p file market protocol for payment [Zuo and Li 2005] (which we shall refer to as Zuo-Li protocol from now on). We also propose two corrections for this protocol, one of which relies on Verifiable and Recoverable Encryption Signatures (VRES) [A. Nenadic and Goble 2005]. We assume familiarity with fair exchange protocols and the strand spaces method[1].

This paper is organised as follows: in Section 2, we describe the common notation used in the remaining sections; in Section 3, we describe the concept of the file market proposed in [Zuo and Li 2005]; in Section 4 we use our adaptation of the strand apaces method, as proposed in [Piva et al. 2006], to trace four attacks to their system; in Section 5 we show our proposed modifications to the protocol and finally, Section 6 accounts for results and conclusions. The reader can refer to Appendix A for the general forms of the theorems used in Section 4.

2. Notation

In this section we describe the common notation used throughout the text. Items i-xi are commonly used in the study of cryptographic protocols and strand spaces, while items xii-xx were introduced in [Zuo and Li 2005]. A user registered in the market is a $peer$. During exchanges, peers are refered to as $parties$.

i. A and B are the parties in the exchange – the Provider and the Downloader respectively – and T or TTP is the mutually trusted third party;
ii. X_{evil}: a dishonest party X;
iii. $\text{SIG}_X(M)$: message M, signed with party X's private key (serves as non-repudiation evidence that X transmitted message M at some point);
iv. $\text{SE}_K(M)$: message M, symmetrically encrypted with key K;
v. $\text{E}_X(M)$: message M, asymmetrically encrypted with party X's public key;
vi. $\text{D}_X(M)$: asymmetrical decryption of message M, using party X's private key;
vii. $\text{H}(M)$: the hash of a message M, obtained by applying a collision-free, one-way function H() to message M;
viii. $M \| N$ or M, N: concatenation of two messages, M and N;
ix. $\rightsquigarrow X : L$: message transmition contaning term L to party X;
x. $G_1, G_2, G_3, ...$: alternative representation of terms that are untestable to the party receiving them (the Downloader, in most cases);
xi. $\rightsquigarrow X : omitted$: party X fails to receive the intended message, either because the sender intentionally did not send it, or because X did not recognize the received information as a valid message;
xii. $M \stackrel{?}{=} N$: comparison between messages M and N, which returns $TRUE$ if $M = N$, or $FALSE$ otherwise;

[1] See [Piva et al. 2006] for a more complete description on related work

xiii. AC: the accounting center, an entity responsible for managing accounts from each peer (by subtracting from the account of any given money equivalent to issued cheques and adding money equivalent to earned cheques);
xiv. $MaxNum$: maximum number of cheques that a certain peer can issue;
xv. $MaxValue$: maximum value of any cheque that a certain peer can issue;
xvi. $CurTime$: the current date and time registered by the system clock;
xvii. $CashExp$: expiration time of a certain capital certificate;
xviii. CC_X: capital certificate of peer X;
xix. TS: timestamp;
xx. $FileID$: the file identificator in the system (its name and size, for instance);
xxi. KP: the key piece, a file piece that the Provider randomly chooses to be symmetrically encrypted with a key unknown to the Downloader. This key will be traded during the payment step by a cheque signed by the Downloader;
xxii. $ChequeSN$: the serial number of a given cheque (to be compared to $MaxNum$ by the payee). It reflects the payer's personal counter, which is incremented every time he issues a cheque;
xxiii. $Price$: the price to be payed for the file in a given exchange, in the form of one or more signed cheques.

3. The Zuo-Li p2p file market

We now describe the file market proposed in [Zuo and Li 2005]. In their original work, the authors describe how to construct a hypothetically fair file-exchanging community, based on the p2p (peer to peer) technology. Section 3.1 shows the protocols suggested by the authors for each phase of the exchange, and Section 3.2 gives a more detailed description of the actions performed by each party during the whole process.

3.1. Buying a file: An overview

Files can be obtained in exchange for virtual payment (in the form of signed cheques) accounting for a value. The exchange itself happens in three steps: the Negotiation Step (Protocol 1), in which general parameters are chosen by parties; the Download Step, in which the Downloader obtains all the file pieces; and the Payment Step (Protocol 2, a fair exchange protocol in the sense introduced in [Asokan 1998]), in which the Downloader exchanges a signed cheque for a key to decrypt a particular file piece (aka, the key piece), thus enabling the reassembly of the whole file. If the Downloader fails to receive the correct key, he can invoke the TTP by running an auxiliary subprotocol (Protocol 3), in order to retrieve the key from her.

3.2. Concept description

In this section we detail the proposed file market – from user registration (Setup Phase) to cheque accounting (Accounting Phase), including the steps mentioned in Section 3.1.

1. **System setup:** AC registers a new user as a peer in the system. It chooses its identity (say, X), generates and transmits its private key and issues him a capital certificate $CC_X = \text{SIG}_{AC}(X \parallel ChequeSN \parallel CashExp \parallel MaxValue \parallel MaxNum)$.
2. **Download:** The peer chooses and downloads[2] the file – piece by piece – and pays for it. This phase has two steps:

[2]Files are divided into several pieces, such as in BitTorrent. Each piece has a sequence number.

Protocol 1 Protocol for the Negotiation Step

Setup: After the setup phase, each entity has a capital certificate (CC_X))
Results: B knows the sequence number of the key piece, and is allowed to download

1.	B:	- Chooses which file to download; - Generates and signs $DownREQ = (FileID \parallel B \parallel ChequeSN \parallel TS)$; $\rightsquigarrow A$: $DownREQ \parallel \text{SIG}_B(DownREQ) \parallel CC_B$
2.	A:	- Checks B's signature on $\text{SIG}_B(DownREQ)$ and AC's signature on CC_B; - Checks if $ChequeSN \leq MaxNum$ in CC_B and if $CashExp \geq CurTime$; - Selects a file piece (key piece (KP), with sequence number $DownRES$) to be encrypted; $\rightsquigarrow B$: $DownRES$;

Protocol 2 Protocol for the Payment Step (Fair Exchange)

Setup: After negotiation, Downloader B already downloaded every piece of the desired file (the key piece KP is encrypted with a secret key K)
Results: A has a valid $\text{SIG}_B(C, Z)$ and B has K

1.	A:	- Generates $Z = \text{E}_{TTP}(A, B, K)$ and $C = (A \parallel B \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel \text{H}(KP) \parallel \text{H}(\text{SE}_K(KP)))$; $\rightsquigarrow B$: $C \parallel Z \parallel \text{SIG}_A(C, Z)$
2.	B:	- Checks C; $\rightsquigarrow A$: $\text{SIG}_B(C, Z)$;
3.	A:	- Checks $\text{SIG}_B(C, Z)$; $\rightsquigarrow B$: K;
4.	B:	- Waits for K or timeout; in case of error, runs Protocol 3;

Protocol 3 Subprotocol for the Payment Step (Fair Exchange)

Setup: Downloader B has encountered some problem and has not correctly received K
Results: B has K and A has $SIG_B(C, Z)$

1.	B:	$\rightsquigarrow TTP$: $C \parallel Z \parallel \text{SIG}_A(C, Z) \parallel \text{SIG}_B(C, Z)$
2.	TTP:	- Checks if the first three terms of C are A, B and the TTP's identities; - Computes $D_{TTP}(Z)$ and checks if the result equals $(A, B, *)$; $\rightsquigarrow B$: K; $\rightsquigarrow A$: $\text{SIG}_B(C, Z)$;

(a) **Negotiation:** Downloader B requests a file hosted by Provider A. The request is described as a signed term $DownREQ$ (Protocol 1). B must also provide his capital certificate CC_B, so that A can verify if B still has credit. If all checks succeed, A sends $DownRES$ to B. It contains the sequence number of a random piece of the file (known as key piece, or KP), wich will be encrypted by A with a key K unknown to B. This key will be negotiated during the next step. B is now allowed to download all file pieces (including the encrypted KP).

(b) **Payment:** To reassemble the original file from the downloaded pieces, B must obtain the key K. Only then he will be able to get the key piece KP. A will trade the correct key by a signed cheque from B, which will then be converted by the AC into credit for A. This procedure happens through a fair exchange protocol, presented in Protocols 2 and 3.
Note that, if B fails to receive key K after sending the signed cheque, he can invoke the assistance of a trusted third party (TTP). The third party will require the cheque and some other information from B, as described in Protocol 3. If all checks performed by the TTP succeed, she will send K to B and the signed cheque to A; if any check fails, the TTP stops.

3. **Accounting:** After the exchange is complete, A must obtain the money from the signed cheque. She then sends a quadruplet $(C, Z, \text{SIG}_X(C, Z), K)$ to the accounting center AC, which should reject expired packages and quadruplet replication. AC will subtract the value – according to the cheque's value – from B's account, and add it to A's account. But first, AC checks if: 1) both A and B's identities are in C; 2) $\text{E}_{TTP}(A, B, K) = Z$; 3) B's signature is correct.

4. Attacks on the protocol for payment using the strand spaces model

We now use the method described in Appendix A to show why the proposed protocol for payment fails to achieve both strong fairness and timeliness for the Downloader. Notice that proofs of failure are derived by contradiction. This is a peculiarity of our adaptation of the strand spaces method.

4.1. Traces definition

1. A strand $s_p \in \text{PROV}[X,\ Y,\ TTP,\ K,\ \text{SIG}_Y(C, Z),\ C,\ \text{SIG}_X(C, Z)]$ iff s_p has trace of the form

$$\langle +C \parallel Z \parallel \text{SIG}_X(C, Z),\ -\text{SIG}_Y(C, Z),\ +K\rangle, \text{ where}$$

   i. $X, Y \in T_{name}$ with $X \neq Y$, $TTP \in T_{ttp}$ with T_{name} and T_{ttp} disjoint;
   ii. $C = (X \parallel Y \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel \text{H}(KP) \parallel \text{H}(\text{SE}_K(KP)))$;
   ii. $Z = \text{E}_{TTP}(X, Y, K)$;
   iii. H() is a one-way, collision-free hash function;
   iv. $K \in \mathcal{K}$ is X's initial object with description C; and
   v. $\text{SIG}_Y(C, Z)$ is Y's object with description $\text{SIG}_X(C, Z)$.

   The principal associated with a strand $s_p \in \text{PROV}[\ ]$ is A.

2. A strand $s_d \in \text{DOWN}[X,\ Y,\ TTP,\ K,\ \text{SIG}_Y(C, G_1),\ C,\ \text{SIG}_X(C, G_1)]$ iff s_d has trace of the form

$$\langle -C \parallel G_1 \parallel \text{SIG}_X(C, G_1),\ +\text{SIG}_Y(C, G_1),\ -K\rangle, \text{ where}$$

   i. $X, Y \in T_{name}$ with $X \neq Y$, $TTP \in T_{ttp}$ with T_{name} and T_{ttp} disjoint;
   ii. $C = (X \parallel Y \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel G_2 \parallel \text{H}(G_3))$ with $G_1, G_2, G_3 \in \mathcal{A}$;
   ii. $Z = \text{E}_{TTP}(X, Y, K)$;
   iii. H() is a one-way, collision-free hash function;
   iv. $K \in \mathcal{K}$ is X's initial object with description C; and
   v. $\text{SIG}_Y(C, G_1)$ is Y's object with description $\text{SIG}_X(C, G_1)$.

The principal associated with a strand $s_d \in \text{DOWN}[\ ]$ is B.

At the end of the protocol, the Downloader represented by s_d should be able to verify that $G_1 = Z = \text{E}_{TTP}(X, Y, K)$, with $G_2 = \text{H}(KP)$ and $\text{H}(G_3) = \text{H}(\text{SE}_K(KP))$. But until K is received, he will accept absolutly any term generatable by the algebra $\mathcal{A}$ as a valid Z (because Z is untestable by the Downloader, as its generation requires knowledge of K).

4.1.1. Strong fairness for B

In this section we show that, by failing to satisfy theorem 1, the protocol does not achieve strong fairness to the Downloader. Proof follows by contradiction. The protocol also fails to achieve timeliness for that party, as many untestable terms are required to begin a subprotocol run (see theorem 6 and refer to [Piva et al. 2006] for more details).

Theorem 1. *Let B be a principal associated with a strand $s_B \in$ DOWN[A, B, TTP, $*$, $\text{SIG}_B(C', G_1)$, C', $\text{SIG}_A(C', G_1)$], where:*

I. $C' = (A \,\|\, B \,\|\, TTP \,\|\, ChequeSN \,\|\, Price \,\|\, TS \,\|\, G_2 \,\|\, \text{H}(G_3))$*;*
II. G_3 *was received on the downloading phase as encrypted* KP*.*

Then, if the protocol provides strong fairness for B, one of the following occurs:

1. $s_B \in$ DOWN[A, B, TTP, K, $\text{SIG}_B(C', Z)$, C', $\text{SIG}_A(C', Z)$]*, where*
   (a) $C' = (A \,\|\, B \,\|\, TTP \,\|\, ChequeSN \,\|\, Price \,\|\, TS \,\|\, G_2 \,\|\, \text{H}(G_3))$*;*
   (b) $G_2 = \text{H}(KP)$*;*
   (c) $G_3 = \text{SE}_K(KP)$*;*
   (d) $Z = \text{E}_{TTP}(A, B, K)$.
2. $\forall s_X \in$ PROV[$*$, B, $*$, $*$, $*$, C', $\text{SIG}_X(C', G_1)$]*, then*
   $s_X \notin$ PROV[$*$, B, $*$, $*$, $\text{SIG}_B(C', G_1)$, C', $\text{SIG}_X(C', G_1)$]*, with:*
   (a) $X \in T_{name}$
   (b) $G_1 = \text{E}_{TTP}(A, B, G_4)$ [3].

Proof: Supose that the comunication channel between principals is resilient. For a given bundle $\mathcal{C}$, if $\exists s_B \in$ DOWN[A, B, TTP, $*$, $\text{SIG}_B(C', G_1)$, C', $\text{SIG}_A(C', G_1)$], then $\exists s_A \in$ PROV[A, B, $*$, $\text{SIG}_B(C', G_1)$, $*$, C', $\text{SIG}_A(C', G_1)$] with $\mathcal{C}-$height ≥ 2. The following cases are possible: $\mathcal{C}-\text{height}(s_A) = 1$ (no item is transmitted, no harm done); $\mathcal{C}-\text{height}(s_A) = 2$ (see below); and $\mathcal{C}-\text{height}(s_A) = 3$ (effectiveness if message 3 contains K, same as $\mathcal{C}-\text{height}(s_A) = 2$ otherwise).

Say that s_A has $\mathcal{C}-$height $= 2$. As $\mathcal{C}-\text{height}(s_A) < 3$, message 3 is never transmitted, so B must invoke the auxiliary subprotocol to complete the exchange. Then $\exists s_{TTP} \in$

[3]Note that it is not necessary that $G_1 = Z$, nor that $C' = C = (A \,\|\, B \,\|\, TTP \,\|\, ChequeSN \,\|\, Price \,\|\, TS \,\|\, \text{H}(KP) \,\|\, \text{H}(\text{SE}_K(KP)))$ for $\text{SIG}_B(C', G_1)$ to be valid. This is a consequence of the checks performed by the AC during the accounting phase of the protocol; it does not check if G_4 is a valid key, neither if G_2 and G_3 are what they are supposed to be (information on KP would be required), but it checks if the two first terms of the triple (A, B, G_4) are A's and B's identities, and if G_1 is an encryption of that triple using the TTP's public key.

TRUST$[A,\ B,\ TTP,\ *,\ \text{SIG}_B(C', G_1),\ C',\ \text{SIG}_A(C', G_1)]$ with $\mathcal{C}-$height ≥ 1. Because the TTP is trustworthy, she will always behave honestly and try to complete the subprotocol run.

After receiving B's request for intervention, the TTP will compute $\text{D}_{TTP}(G_1)$. This will result in one of the following situations:

1. **$\text{D}_{TTP}(G_1)$ fails[4]:** The TTP fails to obtain G_4 (which it expects to be the key K), and can not provide fairness to B. This is represented by Attack 1.

Attack 1 Timeliness attack exploring untestability of Z ($Z_{bogus} \neq E_{TTP}(*)$)

Assumptions: B obtained $\text{SE}_K(KP))$ during the negotiation phase
Steps: Steps 1-3 represent a particular run of the payment protocol, while steps 4-5 represent an attempt of B to run the subprotocol
Results: B is unable to successfully invoke TTP using Protocol 3

1.	A_{evil}:	- Generates C' and Z_{bogus}, with $Z_{bogus} \neq E_{TTP}(*)$, and $C' = (A \parallel B \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel G_2 \parallel \text{H}(G_3))$; $\rightsquigarrow B$: $C' \parallel Z_{bogus} \parallel \text{SIG}_A(C', Z_{bogus})$
2.	B:	- Computes $\text{H}(\text{SE}_K(KP)))$ and checks if it equals to $H(G_3)$; - Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$ and A's signature; - Signs (C', Z_{bogus}); $\rightsquigarrow A_{evil}$: $\text{SIG}_B(C', Z_{bogus})$;
3.	A_{evil}:	- Checks B's signature; $\rightsquigarrow B$: $ommited$;
4.	B:	- Waits for K until timeout; $\rightsquigarrow TTP$: $C' \parallel Z_{bogus} \parallel \text{SIG}_A(C', Z_{bogus}) \parallel \text{SIG}_B(C', Z_{bogus})$;
4.	TTP:	- Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$; - Computes $\text{D}_{TTP}(Z_{bogus})$ and obtains (W_1, W_2, W_3); - Checks that $W_1 \neq A$ and $W_2 \neq B$, and stops.

This situation happens because even though $C' = (A \parallel B \parallel TTP \parallel ...)$ appears to be a proof that A and B agree over the TTP's identity, the protocol does not guarantee that A will use that TTP's public key to generate Z (which is untestable to B). This should already be clear from the fact that $s_A \in$ PROV$[A,\ B,\ *,\ \text{SIG}_B(C', G_1),\ *,\ C',\ \text{SIG}_A(C', G_1)]$ is not limited by the TTP's identity (i.e., the third parameter is a "*").

2. **$\text{D}_{TTP}(G_1)$ succeeds:** TTP reads the triple (t_1, t_2, t_3). If $t_1 \neq A$ or $t_2 \neq B$, TTP also stops, which results in Attacks 2 and 3 (notice that those attacks are very similar to the previous one, in the sense that they also rely on a specially constructed Z_{bogus}). If $(t_1, t_2, t_3) = (A, B, K)$, then $G_1 = \text{E}_{TTP}(A, B, K)$ and we have both $s_B \in$ DOWN$[A,\ B,\ TTP,\ K,\ \text{SIG}_B(C', Z),\ C',\ \text{SIG}_A(C', Z)]$

[4]By "fails" we mean that the decryption results in useless data or garbage, in the sense that G_1 is not an encryption with the TTP's public key. Attacks 2 and 3 are actually subcases of Attack 1, but we believe that the case-by-case approach is far clearer than the straightforward one.

and $s_A \in$ PROV$[A,\ B,\ TTP,\ \text{SIG}_B(C', Z),\ *,\ C',\ \text{SIG}_A(C', Z)]$, with $C' = (A \parallel B \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel G_2 \parallel \text{H}(G_3))$. Notice that now B is finally able to decrypt G_3, but nothing guarantees that this decryption will succeed and B will get KP. This is another flaw of the market proposed by the authors.

Attack 2 Timeliness attack exploring untestability of Z ($Z_{bogus} = E_{TTP}(A, W, G_4)$)

Assumptions: B obtained $\text{SE}_K(KP))$ during the negotiation phase
Steps: Steps 1-3 represent a particular run of the payment protocol, while steps 4-5 represent an attempt of B to run the subprotocol
Results: B is unable to successfully invoke TTP using Protocol 3

1.	A_{evil}:	- Generates C' and Z_{bogus}, with $Z_{bogus} = E_{TTP}(A, W, G_4)$, $W \neq B$ and $C' = (A \parallel B \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel G_2 \parallel \text{H}(G_3))$; - Generates $Z_{bogus} = E_{TTP}(A, W, G_4)$, with $W \neq B$; $\rightsquigarrow B$: $C' \parallel Z_{bogus} \parallel \text{SIG}_A(C', Z_{bogus})$
2.	B:	- Computes $\text{H}(\text{SE}_K(KP)))$ and checks if it equals to $H(G_3)$; - Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$ and A's signature; - Signs (C', Z_{bogus}); $\rightsquigarrow A_{evil}$: $\text{SIG}_B(C', Z_{bogus})$;
3.	A_{evil}:	- Checks B's signature; $\rightsquigarrow B$: $ommited$;
4.	B:	- Waits for K until timeout; $\rightsquigarrow TTP$: $C' \parallel Z_{bogus} \parallel \text{SIG}_A(C', Z_{bogus}) \parallel \text{SIG}_B(C', Z_{bogus})$;
5.	TTP:	- Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$; - Computes $\text{D}_{TTP}(Z_{bogus})$ and obtains (W, B, G_4); - Checks that $W \neq B$ and stops.

If $t_3 \neq K$, on the other hand, we have another subcase of the above situation, now with $Z_{bogus} = \text{E}_{TTP}(A, B, K_{bogus})$. All the above problems keep B from getting K, but do not provide A with a valid Z' (one which would be recognized by the AC). Attack 4 details this vulnerability, which undermines strong fairness.
Notice that $Z_{bogus} = \text{E}_{TTP}(A, B, K_{bogus})$ is a valid Z', as it passes all the checks performed by the AC. This implies that a dishonest Provider A_{evil} can successfully cash B's payment $\text{SIG}_B(C', Z_{bogus})$ without providing him with the right key to KP. The honest Downloader B would end up without the file, while A would receive the money from B's account.

4.2. Discussion and correction

The last attack only succeeds because AC cannot verify that the key delivered to the Downloader is the one used to encrypt the key piece KP. This allows A to encrypt the key piece with one key and transmit another key within message 2 of the protocol for payment. Notice that by sending $Z_{bogus} = \text{E}_{TTP}(A, B, K_{bogus})$ to B, A completely invalidates the TTP's presence. The Downloader will try to decrypt $\text{SE}_K(KP)$ with K_{bogus}, and will fail to obtain KP. As K_{bogus} was provided by the TTP, B will no longer have any reason to trust the TTP. The Provider, on the other hand, will manage to cash B's signed cheque. He generates the quadruplet $(C', Z_{bogus}, \text{SIG}_B(C', Z_{bogus}), K_{bogus})$.

Attack 3 Timeliness attack exploring untestability of Z ($Z_{bogus} = E_{TTP}(W, B, G_4)$)

Assumptions: B obtained $\mathrm{SE}_K(KP))$ during the negotiation phase
Steps: Steps 1-3 represent a particular run of the payment protocol, while steps 4-5 represent an attempt of B to run the subprotocol
Results: B is unable to successfully invoke TTP using Protocol 3

1.	A_{evil}:	- Generates C' and Z_{bogus}, with $Z_{bogus} = E_{TTP}(W, B, G_4)$, $W \neq A$ and $C' = (A \parallel B \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel G_2 \parallel \mathrm{H}(G_3))$; $\rightsquigarrow B$: $C' \parallel Z_{bogus} \parallel \mathrm{SIG}_A(C', Z_{bogus})$
2.	B:	- Computes $\mathrm{H}(\mathrm{SE}_K(KP)))$ and checks if it equals to $H(G_3)$; - Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$ and A's signature; - Signs (C', Z_{bogus}); $\rightsquigarrow A_{evil}$: $\mathrm{SIG}_B(C', Z_{bogus})$;
3.	A_{evil}:	- Checks B's signature; $\rightsquigarrow B$: $ommited$;
4.	B:	- Waits for K until timeout; $\rightsquigarrow TTP$: $C' \parallel Z_{bogus} \parallel \mathrm{SIG}_A(C', Z_{bogus}) \parallel \mathrm{SIG}_B(C', Z_{bogus})$;
5.	TTP:	- Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$; - Computes $\mathrm{D}_{TTP}(Z_{bogus})$ and obtains (W, B, G_4); - Checks that $W \neq A$ and stops.

The three checks performed by AC (item 3 of Section 3.2) succeed, and so AC accepts the cheque as valid.

5. Corrected version for the payment protocol

In the last section we have shown that the original protocol for payment fails to achieve both timeliness and fairness for B. All suggested attacks were based on the fact that the Downloader is not able to test (as defined on Appendix A, Definition A.2.6) term Z, which is necessary for starting a subprotocol run with the TTP. In particular, Attacks 1, 2 and 3 undermine Timeliness, and rely on the non-testability of the first two values of Z – which represent the Provider and Downloader's identities, respectively. One way to overcome this weakness is to allow the Downloader to test the first two values of Z, by breaking it into two pieces. This idea originated a corrected version of the protocol, illustrated by Protocols 4 and 5

Notice that now the Downloader is able to test if the identities in Z_1 are correct, by encrypting (A, B) and comparing the ciphertext with the received Z_1. If the check succeeds, B can be sure that, in case of any problems on later steps of the exchange, he will be able to invoke the TTP by running the subprotocol. Although making terms testable usually solve Timeliness problems, this does not work with fairness, as K can still be different from the key used to encrypt KP (and the TTP does not check its correctness). Moreover, if the Downloader knew the value of K (for the purpose of generating $\mathrm{E}_{TTP}(K)$ and comparing it with Z_2), he would be able to simply stop the exchange without sending the signed cheque to the Provider (which would represent unfairness for the Provider).

Attack 4 Fairness attack exploring the fact that the AC does not check K

Assumptions: B obtained $\mathrm{SE}_K(KP))$ during the negotiation phase
Steps: Steps 1-3 represent a particular run of the payment protocol, while steps 4-6 represent an attempt of B to run the subprotocol
Results: B is unable to obtain the correct key K, while A obtains a valid cheque signed by B

1.	A_{evil}:	- Generates both $Z_{bogus} = E_{TTP}(A, B, K_{bogus})$, with $K_{bogus} \neq K$, and $C' = (A \parallel B \parallel TTP \parallel ChequeSN \parallel Price \parallel TS \parallel G_2 \parallel \mathrm{H}(G_3))$; $\rightsquigarrow B$: $C' \parallel Z_{bogus} \parallel \mathrm{SIG}_A(C', Z_{bogus})$
2.	B:	- Computes $\mathrm{H}(\mathrm{SE}_K(KP)))$ and checks if it equals to $H(G_3)$; - Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$ and A's signature; - Signs (C', Z_{bogus}); $\rightsquigarrow A_{evil}$: $\mathrm{SIG}_B(C', Z_{bogus})$;
3.	A_{evil}:	- Checks B's signature; $\rightsquigarrow B$: $ommited$;
4.	B:	- Waits for K until timeout; $\rightsquigarrow TTP$: $C' \parallel Z_{bogus} \parallel \mathrm{SIG}_A(C', Z_{bogus}) \parallel \mathrm{SIG}_B(C', Z_{bogus})$;
4.	TTP:	- Checks if $C' = (A \parallel B \parallel TTP \parallel ...)$; - Computes $\mathrm{D}_{TTP}(Z_{bogus})$ and obtains (A, B, K_{bogus}); - Checks indentities of A and B; $\rightsquigarrow B$: K_{bogus}; $\rightsquigarrow A_{evil}$: $\mathrm{SIG}_B(C', Z_{bogus})$

Protocol 4 Proposed protocol for payment step (providing Timeliness)

Setup: After negotiation, Downloader B already downloaded every piece of the desired file (except for one piece, the key piece KP)
Results: A has a valid $\mathrm{SIG}_B(C, Z_1, Z_2)$ and B has the sent K

1.	A:	- Generates $Z_1 = \mathrm{E}_{TTP}(A, B)$ and $Z_2 = \mathrm{E}_{TTP}(K)$; $\rightsquigarrow B$: $C \parallel Z_1 \parallel Z_2 \parallel \mathrm{SIG}_A(C, Z_1, Z_2)$
2.	B:	- Checks C and if $Z_1 = \mathrm{E}_{TTP}(A, B)$; $\rightsquigarrow A$: $\mathrm{SIG}_B(C, Z_1, Z_2)$;
3.	A:	- Checks $\mathrm{SIG}_B(C, Z_1, Z_2)$; $\rightsquigarrow B$: K;
4.	B:	- Waits for K or timeout; in case of error, runs the auxiliary subprotocol for payment;

One solution for this unfairness is the use of Verifiable and Recoverable Encryption Signatures (VRES). The use of VRES implies in four phases: i.initialization, ii.VRES generation, iii.VRES verification and, iv.VRES recovery. One implementation using DSA signatures (DSA-CEGD) was proposed in [A. Nenadic and Goble 2005], where an encryption key is exchanged for a signature. In the initialization phase, the interaction of

Protocol 5 Proposed subprotocol for payment step (providing Timeliness)

Setup: Downloader B has not correctly received the expected K
Results: B has the sent K and A has $SIG_B(C, Z_1, Z_2)$

1.	B:	$\rightsquigarrow TTP$: $C \parallel Z_1 \parallel Z_2 \parallel \text{SIG}_A(C, Z_1, Z_2) \parallel \text{SIG}_B(C, Z_1, Z_2)$
2.	TTP:	- Checks if the first three terms of C are A, B and the TTP's identities; - Computes $D_{TTP}(Z_1)$ and checks for A and B's identities; $\rightsquigarrow B$: K; $\rightsquigarrow A$: $\text{SIG}_B(C, Z_1, Z_2)$;

the TTP and each party generates the certificates C_{BT} and C_{AT}[5]. The DSA-CEGD protocol ensures several properties for exchange of e-goods: (i) strong fairness; (ii) non-repudiation; (iii) content assurance (the receiver can verify during execution that the item to be received will indeed match the description); and (iv) offline $STTP$ (semi-trusted and transparent third party). The payment protocol (Protocol 6) and recovery subprotocol (Protocol 7) are the following:

Protocol 6 Proposed protocol for payment step (providing Fairness)

Setup: certificates C_{BT} and C_{AT}
Results: A has a valid $\text{SIG}_B(C, Z)$ and B has K

1.	A:	generates $Z = \text{E}_{TTP}(A, B, K)$; $\rightsquigarrow B$: $C \parallel Z \parallel C_{AT} \parallel \text{SIG}_A(C, Z)$
2.	B:	checks C_{AT}, $\text{SIG}_A(C, Z)$ and the TTP's signature on items; $\rightsquigarrow A$: $\text{VRES}_B(C, Z) \parallel C_{BT}$;
3.	A:	checks $\text{VRES}_B(C, Z)$ and C_{BT}; $\rightsquigarrow B$: K;
4.	B:	waits for K or timeout $\rightsquigarrow A$: $\text{SIG}_B(C, Z)$;
5.	A:	checks $\text{SIG}_B(C, Z)$; in case of error, runs the auxiliary subprotocol for recovery of B's signature.

Protocol 7 Proposed recovery subprotocol for payment step (providing Fairness)

Setup: Downloader B has encountered some problem and has not correctly received K
Results: B has K and A has $SIG_B(C, Z)$

1.	B:	$\rightsquigarrow TTP$: $C_{BT} \parallel C_{AT} \parallel \text{VRES}_B(C, Z) \parallel K$
2.	TTP:	$\rightsquigarrow B$: K; $\rightsquigarrow A$: $\text{SIG}_B(C, Z)$;

6. Results and conclusions

Using the parameterization proposed in [Piva et al. 2006], we provide a formal proof that the protocol for payment suggested in [Zuo and Li 2005] is not fair. Our verifi-

[5] $CertDa$ in [A. Nenadic and Goble 2005].

cation shows that the intended optimistic two-party fair exchange protocol fails to achieve both strong fairness and timeliness – two of the desirable fair exchange properties proposed by [Asokan 1998]. We trace four potential attacks on that protocol, one of which allows a dishonest Provider to account a cheque even when the Downloader did not obtain the desired file.

Our results show that, although requiring deep protocol understanding and careful enunciation of theorems for each verification, our method produces insightful results and details regarding security flaws, possible attacks and eventual corrections. The theorems must still be handwritten, even though we believe that part of the process can be automated.

References

A. Nenadic, N. Zhang, Q. S. and Goble, C. (2005). DSA-based verifiable and recoverable encryption of signatures and its application in certified e-goods delivery. In *EEE '05: Proceedings of IEEE Conference on e-Technology, e-Commerce and e-Service*. IEEE Computer Society.

Asokan, A. (1998). *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo.

Guttman, J. D. and Thayer, F. J. (2002). Authentication tests and the structure of bundles. *Theor. Comput. Sci.*, 283(2):333–380.

Piva, F. R., Monteiro, J. R. M., Devegili, A. J., and Dahab, R. (2006). Applying strand spaces to certified delivery proofs. In *Anais do IV SBSeg, Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.

Thayer, F. J., Herzog, J. C., and Guttman, J. D. (1999a). Mixed strand spaces. *I Computer Security Foundations Workshop, 1999*, pages 72–82.

Thayer, F. J., Herzog, J. C., and Guttman, J. D. (1999b). Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2–3):191–230.

Zuo, M. and Li, J. (2005). Constructing fair-exchange p2p file market. In *Proceedings of the 4th International Conference on Grid and Cooperative Computing*, pages 941–946.

A. Fair exchange protocols in the strand spaces method

This appendix briefly describes how the strand spaces method can be adapted to allow formal verification of two-party optimistic fair exchange protocols, as introduced in [Piva et al. 2006].

A.1. Fair exchange roles with general parameters

Fair exchange properties [Asokan 1998] can be mapped to the strand spaces method in the same way that authentication properties do. In strand spaces, demonstrations are based in theorem proving through derivation of trace parameters. Fair exchange properties can also be verified in this manner.

Two-party fair exchange protocols usually involve three roles: an initiator, the principal who starts the protocol; a responder, the principal who is initially contacted by the initiator; and a trusted third party (TTP), which is only invoked if the initiator or the responder decides to abandon the exchange before the end of the main protocol.

Although the parameters that compose a regular strand definition may vary from one protocol to another, there are some parameters common to fair exchange protocols. Most regular trace definitions will have the following parameters:

$$[X, Y, T, o, o', d, d', C, F],$$

where X, Y and T are the initiator, the responder and the TTP's identities respectively; o is the initiator's object, which shall be given to the responder; o' the responder's object, which shall be given to the initiator; d is the description of o, the same as $\text{DESC}(o)$. It is an information that the responder will use as a guarantee that o is what he expects it to be; d' is the description of o', the same as $\text{DESC}(o')$. It is an information that the initiator will use as guarantee that o' is what he expects it to be; C is the cancellation token, issued by the TTP to the caller of the cancel protocol; and F is the finishing token, issued by the TTP to the caller of the finish protocol.

These parameters are generally necessary during fair exchange protocol verification. Note that some of them may not matter in some moments (C and F are ignored during effectiveness analysis), but it is common practice to represent them as *'s rather than simply ignoring them.

A.2. Fair exchange properties in the strand spaces method

In this section we present each fair exchange property as described in [Asokan 1998] along with its representation as a general strand spaces theorem. Notice that for a protocol to achieve any of these properties it is necessary that it achieves that property for each principal interested in the exchange. Here we present each theorem on behalf of the initiator. During the complete verification of the property, analogous theorems must be devised for the responder as well.

A.2.1. Effectiveness

Suppose a player A behaves correctly. If player B also behaves correctly, and both A and B do not want to abandon the exchange, then when the protocol is completed, A has o' such that $\text{DESC}(o') = d'$.

Theorem 2. *Let A be a principal associated with a strand $s_A \in \text{INIT}[A, *, T, o, *, d, d', *, *]$ and B be a principal associated with a strand $s_B \in \text{RESP}[*, B, T, *, o', d, d', *, *]$. If both A and B do not want to abandon the exchange, then $s_A \in \text{INIT}[A, *, T, o, o', d, d', *, *]$ and $s_B \in \text{RESP}[*, B, T, o, o', d, d', *, *]$, where $d = \text{DESC}(o)$ and $d' = \text{DESC}(o')$, and d and d' are the respective descriptions of A's starting object and B's starting object.*

A.2.2. Strong fairness

Supose a player A behaves correctly. Then, when the protocol is completed, either A has o' such that $\text{DESC}(o') = d'$, or B has gained no additonal information about o.

Theorem 3. *Let A be a principal associated with a strand $s_A \in \text{INIT}[A, B, T, o, *, d, d', *, *]$. Then either $s_A \in \text{INIT}[A, B, T, o, o', d, d', *, *]$ with $d' = \text{DESC}(o')$ or $s_B \notin (\text{RESP}[*, B, *, o, o', d, d', *, *] \cup \text{INIT}[B, *, *, *, o, *, d, *, *])$, where $d = \text{DESC}(o)$.*

A.2.3. Weak fairness

Suppose a player A behaves correctly. Then, when the protocol is completed, either A has o' such that $\text{DESC}(o') = d'$, or B has gained no additonal information about o, or A can prove to an arbiter that B has received (or *can* still receive) o such that $d = \text{DESC}(o)$, without any further intervention from A.

Theorem 4. *Let A be a principal associated with a strand $s_A \in \text{INIT}[A, B, T, o, *, d, d', *, *]$. Then either theorem 3 holds or $\exists s_B \in (\text{RESP}[*, B, *, o, o', d, d', *, *] \cup \text{INIT}[B, *, *, *, o, *, d, *, *])$, where $d = \text{DESC}(o)$.*

A.2.4. Non-repudiability

Suppose a player A behaves correctly. Then, after a effective exchange (i.e., A has received o' at the end of the exchange), A will be able to prove

- **Non-repudiability of origin**: that o' originated from B, and
- **Non-repudiability of receipt**: that B received o.

Theorem 5. *Let A be a principal associated with a strand $s_A \in \text{INIT}[A, B, T, o, o', d, d', *, *]$ with $d' = \text{DESC}(o')$ and $d = \text{DESC}(o)$. Then:*

- ***of origin:*** *$\exists s_B \in (\text{RESP}[*, B, *, *, o', *, d', *, *] \cup \text{INIT}[B, *, *, o', *, *, d, *, *])$, with $d' = \text{DESC}(o')$.*
- ***of receipt:*** *$\exists s_B \in \text{RESP}[*, B, *, o, *, d, *, *, *] \cup \text{INIT}[B, *, *, *, o, *, d, *, *])$, with $d = \text{DESC}(o)$.*

A.2.5. Verifiability of TTP

Assuming that the third party T can be forced to eventually send a valid reply to every request, this property requires that if T misbehaves, resulting in the loss of fairness for A, then A can prove the misbehaviour of T to an arbiter (or verifier) in an external dispute. In other words, each of the other players has a weak fairness guarantee even in the case of a misbehaving TTP.

A.2.6. Timeliness

Suppose a player A behaves correctly. Then A can be sure that the protocol will be completed at a certain point in time. At completion, the state of the exchange as of that point is either final or any change to the state will not degrade the level of fairness achieved by A so far.

Definition An item i is *testable* by a principal X if and only if X can check i's validity by doing some computation (by reconstructing i from other testable items, by decrypting i with a known key, etc). If an item is not testable by X, we say it is *untestable* by X.

Theorem 6. *A protocol achieves $timeliness$ for principal A if and only if every item A needs to provide to the TTP in a subprotocol call is testable by A.*