Applying Strand Spaces to Certified Delivery Proofs

Fabio R. Piva[1], **José R. M. Monteiro**[1,2], **Augusto J. Devegili**[1], **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, monteiro, augusto, rdahab}@ic.unicamp.br

Abstract. *Although fair exchange protocols are being widely implemented, there are few formal methods able to verify them. This work introduces the strand spaces method for verifying certified mail delivery protocols, a subclass of fair exchange protocols. Three fair exchange properties are verified: effectiveness, verifiability of TTP and timeliness. For effectiveness and verifiability we used the FPH protocol [Ferrer-Gomila et al. 2000]; for timeliness we use the ZDB protocol [Zhou et al. 1999]. We show that strand spaces can be applied to fair exchange protocols, and present an additional attack to the FPH protocol which was not previously reported.*

1. Introduction

Digital signatures can be used to provide non-repudiation in a variety of well known cryptographic protocols. However, since fairness is more difficult to achieve, it must be provided by fair exchange protocols. These protocols provide communicating parties with assurance regarding the outcome of the exchange: each party receives the item it expects if and only if the other party also gets his or hers. Exchanges are typical of distributed system environments, where negotiations are carried over insecure channels, between mutually trusting parties (except, perhaps, for byzantine-like agreements). Solutions comprise protocols based on a trusted third party (TTP) with varying degrees of involvement. The role of a TTP in an optimistic protocol is restricted to resolving conflicts between the parties, as opposed to being involved in every communication between them, thus reducing the occurrence of bottlenecks involving the TTP, resulting in much greater efficiency. Although Asokan [Asokan 1998] has shown protocols for exchange of generic items, there are fair exchange protocols for certified mail delivery, contract signing and electronic payments. In this work we focus on certified mail delivery.

According to Asokan [Asokan 1998] and Garay et al. [Garay et al. 1999], fair exchange protocols have the following properties: (i) *effectiveness:* the protocol actually exchanges a message for a receipt; (ii) *fairness:* as defined above; (iii) *timeliness:* the parties are guaranteed to complete their exchange in a finite amount of time, even in the presence of exceptions; (iv) *non-repudiation* of the actions of each party (the sender and the recipient); (v) *verifiability of the TTP:* the actions of a TTP may be checked and audited by independent sources; and (vi) *abuse freeness:* if the protocol is executed unsuccessfully, none of the two parties can show the validity of intermediate results to others.

In early protocols, messages were forwarded by the TTP. In 1997, Zhou and Gollmann [Zhou and Gollmann 1997] proposed a solution where the TTP acts as a lightweight

notary. The encrypted message goes directly to the receiver, while the TTP passes only a short-term key when the sender fails to do so. This protocol evolved to an optimistic version [Zhou et al. 1999]. In 2000, Ferrer-Gomila, Payeras-Capellà and Huguet i Rotger [Ferrer-Gomila et al. 2000] presented a new version of Asokan's protocol, which uses the same strategy as Zhou and Gollmann, i.e., which replaces the expected item with another, previously agreed, item: the decrypting key for the message.

Although there has been some recent work on formal verification methods for fair exchange protocols, like [Chadha et al. 2001, Chadha et al. 2004, Mukhamedov et al. 2005], protocol designers usually employ informal techniques to verify fair exchange properties. However, informal techniques may lead to incomplete or flawed verifications, as this paper shows. The verification of fair exchange properties can be made less error prone if formal methods are used. The strand spaces method [Thayer et al. 1999b] allows one to represent cryptographic protocols and prove security properties. By the association of strands to protocol entities and analysing behavioural traces expected for those entities, properties can be proved, such as ambiguities in generation of these traces. Initially, the strand spaces method was designed for a single authentication protocol execution. Later, an adaptation was proposed in [Thayer et al. 1999a]), which accounts parallel executions of the protocol but forbids a term generated in a parallel execution to be used in the main protocol run (which is exactly what occurs in fair exchange protocols). In [Guttman and Thayer 2002], authentication tests are introduced. By the use of test components, it is possible to determine ambiguities in the origin of terms in entity traces.

In [Mukhamedov et al. 2005] the authors use the strand spaces method to provide a formal proof for their corrected version of a flawed multi-party fair exchange protocol, although not considering the multi-protocol nature of fair exchange protocols. Our work applies the notions of authentication tests to prove properties in complete fair exchange protocols – considering not only the main exchange protocol, but also its components finish and cancel protocols. We use the strand spaces method to analyse the effectiveness and non-verifiability of the TTP for the Ferrer-Gomila (FPH) protocol and non-timeliness for the Zhou et al. (ZDB) protocol. We refer the reader to [Thayer et al. 1999b, Thayer et al. 1999a, Guttman and Thayer 2002] for background on the strand spaces method and [Asokan 1998] on fair exchange protocols.

This paper is organised as follows: in Section 2, we describe the common notation used in the remaining sections; in Section 3, we show how strand spaces are adapted to the verification of fair exchange protocols; in Section 4 we present the FPH protocol, notations and analysis; in Section 5 we do the same for the ZDB protocol; and the last section contains results and conclusions of this work.

2. Common notation

In this section we describe the common notation used throughout the text.

i. A and B are the parties in the exchange and T or TTP is the trusted third party (TTP);
ii. M: message from A to be certifiably delivered to B;
iii. K: symmetric key generated by A;
iv. M, N or $M\ N$: concatenation of two messages, M and N;
v. $PR_X(G)$/$PU_X(G)$: encryption of term G with principal X's private/public key;

vi. $A \rightarrow B : G$: principal A sends a message contaning term G to principal B;

vii. $A \rightarrow B : \frac{G \texttt{ (if C1)}}{G' \texttt{ (if C2) } var := VALUE}$: principal A sends to principal B a message containing term G if condition $C1$ holds. If condition $C2$ holds instead, A sends G' to B and attributes $VALUE$ to variable var;

viii. $|G|_X$: term G signed by a principal X. This can be achieved with assymetric cryptography (in this case, $|G|_X = PR_X(G)$).

3. Fair exchange protocols in the strand spaces method

This section describes how the strand spaces method can be adapted to allow verification of fair exchange protocols. We also describe how fair exchange properties presented in [Asokan 1998] may be described as strand spaces theorems.

3.1. 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 $desc(o)$. 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 $desc(o')$. 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.

3.2. 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.

3.2.1. 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 $desc(o') = d'$.

Theorem 3.1. *Let A be a principal associated with a strand $s_A \in Init[A, *, T, o, *, d, d', *, *]$ and B be a principal associated with a strand $s_B \in Resp[*, B, T, *, o', d, d', *, *]$. If both A and B do not want to abandon the exchange, then $s_A \in Init[A, *, T, o, o', d, d', *, *]$ and $s_B \in Resp[*, B, T, o, o', d, d', *, *]$, where $d = desc(o)$ and $d' = desc(o')$, d and d' are the respective descriptions of A's starting object and B's starting object.*

3.2.2. Strong fairness

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

Theorem 3.2. *Let A be a principal associated with a strand $s_A \in Init[A, B, T, o, *, d, d', *, *]$. Then either $s_A \in Init[A, B, T, o, o', d, d', *, *]$ with $d' = desc(o')$ or $s_B \notin (Resp[*, B, *, o, o', d, d', *, *] \cup Init[B, *, *, *, o, *, d, *, *])$, where $d = desc(o)$.*

3.2.3. Weak fairness

Suppose a player A behaves correctly. Then when the protocol is completed, either A has o' such that $desc(o') = d'$, 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 $d = desc(o)$, without any further intervention from A.

Theorem 3.3. *Let A be a principal associated with a strand $s_A \in Init[A, B, T, o, *, d, d', *, *]$. Then either theorem 3.2 holds or $\exists\ s_B \in (Resp[*, B, *, o, o', d, d', *, *] \cup Init[B, *, *, *, o, *, d, *, *])$, where $d = desc(o)$.*

3.2.4. 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 3.4. *Let A be a principal associated with a strand $s_A \in Init[A, B, T, o, o', d, d', *, *]$ with $d' = desc(o')$ and $d = desc(o)$. Then:*

- ***of origin:*** *$\exists\ s_B$ such as $s_B \in (Resp[*, B, *, *, o', *, d', *, *] \cup Init[B, *, *, o', *, *, d, *, *])$, with $d' = desc(o')$.*
- ***of receipt:*** *$\exists s_B$ such as $s_B \in Resp[*, B, *, o, *, d, *, *, *] \cup Init[B, *, *, *, o, *, d, *, *])$, with $d = desc(o)$.*

3.2.5. Verifiability of TTP

Assuming that the third party T can be forced to eventually send a valid reply[1] 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.

3.2.6. 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 3.5. *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*.*

4. The FPH protocol

The FPH protocol, firstly introduced in [Ferrer-Gomila et al. 2000], is a two-party optimistic protocol for certified mail delivery. The items to be exchanged are a message M (originated by the initiator) and its receipt $|h|_B$ (by the responder). Both tokens F and C issued by the TTP are morphologically identical to each other, which results in the attacks described in Section 4.3.2. In this Section we use the following notation:

i. $H(M)$ - the result of applying a collision-free hash function H to message M;
ii. $\{M\}_K$ - encryption of M using a symmetric algorithm with secret key K producing ciphertext $\{M\}_K$; decryption of $\{M\}_K$ is performed with $\{\{M\}_K\}_K$;
iii. $PU_T(K)$ - key K encrypted with the TTP's public-key PU_T;
iv. $|h|_A = |H(H(\{M\}_K), PU_T(K))|_A$ - part of the evidence of non-repudiation of origin of message M for B;
v. $|h|_B = |H(H(\{M\}_K), PU_T(K))|_B$ - part of the evidence of non-repudiation of reception of message M for A;
vi. $|\texttt{key} = K|_A$ - second part of the evidence of non-repudiation of origin for B;
vii. $|\texttt{key} = K|_T$ - an alternative second part of the evidence of non-repudiation of origin for B;
viii. $|H(H(\{M\}_K), PU_T(K), |h|_A)|_A$ - an evidence that A has requested the TTP's intervention;
ix. $|H(H(\{M\}_K), PU_T(K), |h|_A, |h|_B)|_B$ - an evidence that B has requested the TTP's intervention;
x. $|H(|h|_B)|_T$ - the TTP's signature on $|h|_B$ which proves its intervention.

[1] Because of the concept of what a *valid reply* is varies from protocol to protocol, there can not be a general theorem representing this property. See section 4 for an example.

4.1. Protocol description

In this section we describe the three components of the FPH protocol.

1. $A \rightarrow B : \{M\}_K, PU_T(K), |h|_A$ (if exception, B stops)
2. $B \rightarrow A : |h|_B$ (if exception, A runs the cancel protocol)
3. $A \rightarrow B : |\texttt{key = } K|_A$ (if exception, B runs the finish protocol)

Figure 1. Main protocol

1. $B \rightarrow T : H(\{M\}_K), PU_T(K), |h|_A, |h|_B, |H(H(\{M\}_K), PU_T(K), |h|_A, |h|_B)|_B$
2. $T \rightarrow B : \dfrac{|H(\texttt{cancelled, } |h|_B)|_T \texttt{ (if } \textit{cancelled} \texttt{ = TRUE)}}{|\texttt{key = } K|_T \texttt{ (if } \textit{cancelled} \texttt{ = FALSE) } \textit{finished} := TRUE}$

Figure 2. Finish protocol

1. $A \rightarrow T : H(\{M\}_K), PU_T(K), |h|_A, |H(H(\{M\}_K), PU_T(K), |h|_A)|_A$
2. $T \rightarrow A : \dfrac{|h|_B, |H(|h|_B)|_T \texttt{ (if } \textit{finished} \texttt{ = TRUE)}}{|H(\texttt{cancelled, } |h|_A)|_T \texttt{ (if } \textit{finished} \texttt{ = FALSE) } \textit{cancelled} := TRUE}$

Figure 3. Cancel protocol

4.2. Strand spaces representation and regular strands trace definition

In this section we define the traces of regular strands for the main protocol and its two subprotocols (finish and cancel). The strand space Σ may be formed by many possible combinations of those strands.

4.2.1. Main protocol

Figure 4 illustrates the FPH main protocol.

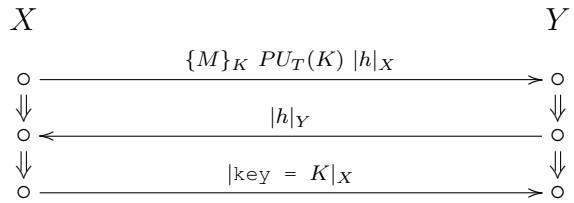

Figure 4. Main protocol

1. A strand $s \in Init[X, Y, *, M, |h|_Y, \{M\}_K, h, *, *, K]$ iff s has trace of the form

$$\langle +\{M\}_K\ PU_T(K)\ |h|_X,\ -|h|_Y,\ +|\texttt{key = } K|_X \rangle$$

where $X, Y \in T_{name}$ with $X \neq Y$, $T \in T_{ttp}$ with T_{name} and T_{ttp} disjoint, $h = H(\{M\}_K, PU_T(K))$ and $H()$ is a one-way, collision-free hash function, M is X's initial object with description $\{M\}_K$, $|h|_Y$ is Y's object with description h and $K \in \mathcal{K}$. The principal associated with a strand $s \in Init$ is A.

2. A strand $s \in Resp[X, Y, *, M, |h|_Y, G, h, *, *, G', K]$ iff s has trace of the form

$$\langle -G\ G'\ |h|_X,\ +|h|_Y,\ -|\texttt{key = } K|_X \rangle$$

The principal associated with a strand $s \in Resp$ is B.

4.2.2. Finish protocol

Figure 5 illustrates the FPH finish protocol.

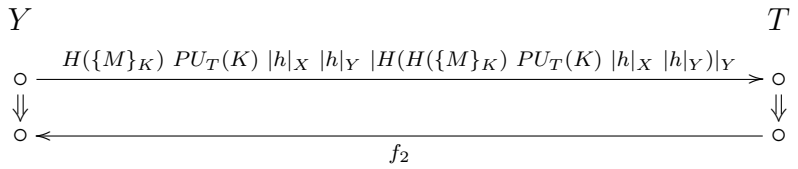

Figure 5. Finish protocol

1. A strand $s \in Resp[X, Y, T, M, |h|_Y, G, h, C_R, F_R, G', K]$ iff s has trace of the form
$$\langle +H(G)\ G'\ |h|_X\ |h|_Y\ |H(H(\{M\}_K)\ PU_T(K)\ |h|_X\ |h|_Y)|_Y,\ -f_2\rangle$$
where $f_2 = C_R = |(H(\texttt{cancelled},\ |h|_Y))|_T$ is a cancel token issued by the TTP to the responder if the protocol has been cancelled and $f_2 = |\texttt{key}\ =\ K|_T$ is issued otherwise. The principal associated with a strand $s \in Resp$ is B.
2. A strand $s \in Serv[X, Y, T, M, |h|_Y, \{M\}_K, h, C_R, F_R, K]$ iff s has trace of the form
$$\langle -H(\{M\}_K)\ PU_T(K)\ |h|_X\ |h|_Y\ |H(H(\{M\}_K)\ PU_T(K)\ |h|_X\ |h|_Y)|_Y,\ +f_2\rangle$$
The principal associated with a strand $s \in Serv$ is T.

4.2.3. Cancel protocol

Figure 6 illustrates the FPH cancel protocol.

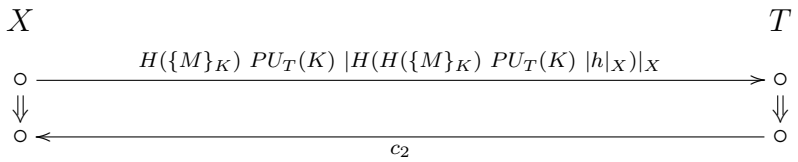

Figure 6. Cancel protocol

1. A strand $s \in Init[X, Y, T, M, G'', \{M\}_K, h, C_I, F_I, K]$ iff s has trace of the form
$$\langle +H(\{M\}_K)\ PU_T(K)\ |H(H(\{M\}_K)\ PU_T(K)\ |h|_X)|_X,\ -c_2\rangle$$
where $c_2 = C_I = |(H(\texttt{cancelled},\ |h|_X))|_T$ is a cancel token issued by the TTP to the initiator if the protocol has not been resolved yet (by execution of the finish protocol) and $c_2 = |h|_Y\ |H(|h|_Y)|_T$ otherwise. The principal associated with a strand $s \in Init$ is A.
2. A strand $s \in Serv[X, Y, T, M, G'', \{M\}_K, h, C_I, F_I, K]$ iff s has trace of the form
$$\langle -H(\{M\}_K)\ PU_T(K)\ |H(H(\{M\}_K)\ PU_T(K)\ |h|_X)|_X,\ +c_2\rangle$$
The principal associated with a strand $s \in Serv$ is T.

The sets *Serv, Resp* and *Init* are pairwise disjoint.

4.3. Verification of the FPH protocol

In this Section we prove that the FPH protocol fails to provide verifiability of the TTP and still achieves effectiveness.

4.3.1. Effectiveness

To prove effectiveness, we demonstrate that if both A and B complete the exchange without trying to abandon the protocol, then it is possible to achieve $s_A \in Init[A, *, T, M, |h|_B, \{M\}_K, h, *, *, K]$ and $s_B \in Resp[*, B, T, M, |h|_B, G, h, *, *, G', K]$, with $G = \{M\}_K$ and $G' = PU_T(K)$.

Theorem 4.1. *Let A be a principal associated with a strand $s_A \in Init[A, *, T, M, *, \{M\}_K, h, *, *, K]$, M, K are uniquely originating in Σ and K is good. Suppose s_A has $C-$height≥ 2 and B's public key is safe*[2]*. Then $s_A \in Init[A, *, T, M, |h|_B, \{M\}_K, h, *, *, K]$.*

Proof. The first message received by A (node $\langle s_A, 2\rangle$) contains term $|h|_B$, which can be interpreted as an unsolicited authentication test [Guttman and Thayer 2002] (h is recently generated by A from the uniquely originated values $\{M\}_K$ and $PU_T(K)$). That means that there is a regular participant B (with $B \neq A$) represented by a strand $s_B \in \Sigma$ in which $|h|_B$ is originated. We must consider two cases for s_B (we will not consider the case in which B can be the TTP, as we consider sets T_{name} and T_{ttp} to be disjoint):

1. $s_B \in Init[B, *, T, M, *, \{M\}_K, h, *, *, K]$ and $|h|_B \in term(\langle s_B, 1\rangle)$. From the unique origination of M, K we conclude that $A = B$, which is a contradiction.
2. $s_B \in Resp[*, B, T, *, |h|_B, G, h, *, *, G', *]$ and $|h|_B = term(\langle s_B, 2\rangle)$. Note that G and G' are untestable by B, which means that he can only retrieve M and check if $G = \{M\}_K$ if he receives a correct K, and he can never check if $G' = PU_T(K)$. But he can check that $h = H(G, G')$. So, under A's point of view, the reception of $|h|_B$ means that B has checked $h = H(G, G')$. But A knows the values of G and G', and can conclude that $s_B \in Resp[*, B, T, *, |h|_B, \{M\}_K, h, *, *, PU_T(K), *]$. By the unique origination of M and the goodness of K, A knows from $\{M\}_K$ that B is running the same exchange as he is, and so $s_A \in Init[A, B, T, M, |h|_B, \{M\}_K, h, *, *, K]$.

Theorem 4.2. *Let B be a principal associated with a strand $s_B \in Resp[*, B, T, X, |h|_B, G, h, *, *, G', *]$. Suppose s_B has $C-$height$= 3$ and A's public key is safe. Then $s_B \in Resp[*, B, T, M, |h|_B, G, h, *, *, G', K]$, with $G = \{M\}_K$.*

Proof. The last message received by B (node $\langle s_B, 3\rangle$) can be interpreted as an unsolicited authentication test [Guttman and Thayer 2002]. That means that there is a regular participant A (with $B \neq A$) represented by a strand $s_A \in \Sigma$ in which $|key = K|_A$ is originated. Only initiator strands have a term with that form, so $s_A \in Init[A, *, *, *, *, *, *, *, *, K]$ (note that B must trust A in the generation of a good key K). Now B can check if the other values are correct. It checks that $G = \{M\}_K$ by decrypting G with K. From the goodness of K, B concludes that $G = \{M\}_K$ was originated by A, and so was h. So $s_A \in Init[A, *, T, M, *, \{M\}_K, h, *, *, K]$, and $s_B \in Resp[A, B, T, M, |h|_B, G, h, *, *, G', K]$, with $G = \{M\}_K$. ■

4.3.2. Verifiability of TTP

We show that the FPH protocol does not provide verifiability of TTP.

[2] By safe we mean only known to B.

Theorem 4.3. *The FPH protocol provides verifiability of TTP in the following circumstances:*

- ***as for non-repudiability of origin:*** $\exists s_B$ *such that* $s_B \in Resp[A, B, T, M, *, G, h, *, *, G', K]$*, with* $G = \{M\}_K, G' = PU_T(K)$ *and* K *good, then* $\nexists\, s_A \in Init[A, *, T, M, *, \{M\}_K, h, C, *, K]$ *with a valid cancellation token* C *(a cancellation token* C *is valid if* $C = |(H(\texttt{cancelled},\ |h|_A))|_T$*).*
- ***as for non-repudiability of receipt:*** $\exists s_A$ *such that* $s_A \in Init[A, B, T, M, |h|_B, \{M\}_K, h, *, *, K]$*, with* $PR_B()$ *safe, then* $\nexists s_B \in Resp[*, B, T, M, *, G, h, C, *, G', K]$ *with* $G = \{M\}_K, G' = PU_T(K)$ *and a valid cancellation token* C *(a cancellation token* C *is valid if* $C = |(H(\texttt{cancelled},\ |h|_B))|_T$*).*

Proof. Suppose that the protocol achieves verifiability of TTP as for non-repudiability of receipt. Then $\exists s_A \in Init[A, B, T, M, |h|_B, \{M\}_K, h, *, *, K]$, and as $PR_B()$ is safe, $|h|_B$ must have been originated in the second node of a responder strand, from a main protocol run, or in the first node in a responder strand, from a finish protocol run (in that case A would be participating as the TTP, which is not possible due to T_{name} and T_{ttp} being disjoint). So we conclude that s_A has $C-$height≥ 2 in a main protocol run.

As we supposed that the protocol achieves verifiability of TTP, there can be no principal B who knows a valid C for the protocol run in which A is involved (that means $C = |(H(\texttt{cancelled},\ |h'|_B))|_{T'}$ with $T' = T$, $h' = h = H(\{M\}_K, PU_T(K))$). If there is a principal B who knows a valid token C, then C would have been originated in one of the following cases:

1. finish protocol run: C was produced by T on an instance of finish protocol, but an instance of the cancel protocol must have been called before by some principal Z. So $s_B \in Resp[*, B, T, *, |h|_B, G, h, C, *, G', *]$ and $\exists s_Z \in Init[Z, *, T, *, *, \{M\}_K, h, C', *, K]$, where $C' = |(H(\texttt{cancelled},\ |h|_Z))|_T$. Note that parameters $\{M\}_K$ and K come from the agreement of B and principal Z associated to s_Z over the values $H(\{M\}_K)$ and $PU_T(K)$, sent on the first message on the cancel protocol run. In that case, by the unique origination of M, either $Z = A$ or $Z = B$.
2. cancel protocol run: C was produced by T on an instance of the cancel protocol, but no instance of the finish protocol can have been called before for the values of T and h. So $s_B \in Init[B, *, T, *, *, \{M\}_K, h, C, *, K]$. ■

The protocol allows both scenarios, which represents failure in achieving verifiability of TTP. In the first case, let $Z = B$. Principal A initiates an exchange with principal B, who fakes an exchange initiated by himself and calls the TTP for cancellation any time after receiving the first message from A. The TTP issues the cancellation token $C' = |(H(\texttt{cancelled},\ |h|_B))|_T = C$ to B and marks the fake exchange as finished. Now B calls the TTP for finishing the non-fake run. The TTP identifies A as the initiator and checks that this run has not been cancelled, issues the key K to B, and marks the non-fake run as finished. Now B can go through the rest of the main protocol run by sending $|h|_B$ to A, or he can simply not send anything to A, which will cause A to call the TTP for cancellation. As the run has been finished by B, A gets $|h|_B$ from the TTP. At the end of the exchange, $s_A \in Init[A, B, T, M, |h|_B, \{M\}_K, h, *, *, K]$ and $s_B \in Resp[*, B, T, M, |h|_B, G, h, C, *, G', K]$ with $G = \{M\}_K, G' = PU_T(K)$ and a

valid cancellation token $C = |(H(\texttt{cancelled},\ |h|_B))|_T$. This violates theorem 4.3 and is exactly the attack described in [Monteiro and Dahab 2002].

If we consider the last scenario, a similar attack would be possible. If B had continued the main protocol run right after the reception of the cancellation token C from the TTP, instead of having called the finish protocol, principal A would also get h_B and B would also have C. To the best of our knowledge, this attack has not yet been reported.

Both attacks are possible because the cancellation tokens issued by the TTP in both subprotocols are very similar. If the identity of the initiator had been inserted in the tokens, none of the described attacks would have been possible (C' and C would be different).

5. The ZDB protocol

The ZDB protocol is a two-party non-repudiation protocol proposed by [Zhou et al. 1999] and already analysed and improved in [Boyd and Kearney 2000, Gürgens et al. 2005]. The exchange happens in two steps: First the initiator exchanges a term C (which is a message M encrypted with a key K initially unknown to the responder) for a receipt EOR_C. Then the initiator exchanges the key K for another receipt EOR_K. These two receipts form the responder's object o'. We use the following notation

i. $H(M, K)$: a one-way hash function applied to message M and key K;
ii. $e_K(M)$ and $d_K(M)$: encryption and decryption of message M with key K;
iii. $C = e_K(M)$: committed ciphertext for message M;
iv. $L = H(M, K)$: label to link C and K;
v. $f_1, f_2, \ldots, f_8$: message tags to indicate the purpose of the respective message;
vi. $PU_T(K)$ encryption of key K with TTP's public key;
vii. $sig_A(W)$: principal A's digital signature on message W with A's private signature key. Note that the plaintext is not recoverable from the signature;
viii. $EOO_C = sig_A(f_1, B, L, C)$: evidence of origin of C;
ix. $EOR_C = sig_B(f_2, A, L, EOO_C)$: evidence of receipt of C;
x. $EOO_K = sig_A(f_3, B, L, K)$: evidence of origin of K;
xi. $EOR_K = sig_B(f_4, A, L, EOO_K)$: evidence of receipt of K;
xii. $sub_K = sig_A(f_5, B, L, K, TTP, EOO_C)$: evidence of submission of K to TTP;
xiii. $con_K = sig_{TTP}(f_6, A, B, L, K)$: evidence of confirmation of K by the TTP;
xiv. $abort = sig_{TTP}(f_8, A, B, L)$: evidence of abortion;

5.1. Protocol description

In this section we describe the three components of the ZDB protocol.

1. $A \rightarrow B : f_1, f_5, B, L, C, T, PU_T(K), EOO_C, sub_K$ (if exception, B stops)
2. $B \rightarrow A : f_2, A, L, EOR_C$ (if exception, A runs the cancel protocol)
3. $A \rightarrow B : f_3, B, L, K, EOO_K$ (if exception, B runs the finish protocol)
4. $B \rightarrow A : f_4, A, L, EOR_K$ (if exception, A runs the finish protocol)

Figure 7. Main protocol

1. $U \rightarrow T : f_1, f_2, f_5, A, B, L, T, PU_T(K), sub_K, EOO_C, EOR_C$
2. $T \rightarrow U : \dfrac{f_8, A, B, L, abort \texttt{ (if aborted = TRUE)}}{f_2, f_6, A, B, L, K, con_K, EOR_C \texttt{ (if aborted = FALSE) } finished := TRUE}$

Figure 8. Finish protocol

1. $A \rightarrow T : f_7, B, L, sig_A(f_7, B, L)$
2. $T \rightarrow A : \dfrac{f_2, f_6, A, B, L, K, con_K, EOR_C \texttt{ (if } \textit{finished} \texttt{ = TRUE)}}{f_8, A, B, L, abort \texttt{ (if } \textit{finished} \texttt{ = FALSE)} \; aborted := TRUE}$

Figure 9. Cancel protocol

5.2. Strand spaces representation and regular strands trace definition

In this section we define the regular strands traces for the three ZDB components.

5.2.1. Main protocol

Figure 10 illustrates the ZDB main protocol.

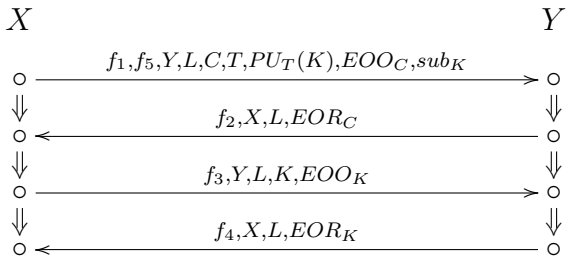

Figure 10. Main protocol

1. A strand $s \in Init[X, Y, T, M, K, EOR_C, EOR_K, C, PU_T(K), EOO_C, EOO_K, *, *, L]$ iff s has trace of the form
   $\langle +\{f_1\ f_5\ Y\ L\ C\ T\ PU_T(K)\ EOO_C\ sub_K,\ -f_2\ X\ L\ EOR_C,\ +f_3\ Y\ L\ K\ EOO_K,\ -f_4\ X\ L\ EOR_C\rangle$
   where $X, Y \in T_{name}$ with $X \neq Y$, $T \in T_{ttp}$ with T_{name} and T_{ttp} disjoint, $K \in \mathcal{K}$. The principal associated with a strand $s \in Init[X, Y, T, M, K, EOR_C, EOR_K, C, PU_T(K), EOO_C, EOO_K, *, *, L]$ is A.
2. A strand $s \in Resp[X, Y, T, M, K, EOR_C, EOR_K, G_2, G_3, EOO_C, EOO_K, *, *, G_1, G_4]$ iff s has trace of the form
   $\langle -\{f_1\ f_5\ Y\ G_1\ G_2\ T\ G_3\ EOO_C\ G_4,\ +f_2\ X\ G_1\ EOR_C,\ -f_3\ Y\ G_1\ K\ G_5,\ +f_4\ X\ G_1\ EOR_C\rangle$
   where $G_1, G_2, G_3, G_4, G_5 \in \mathcal{A}$ (which is the set of all possible terms in Σ, as introduced in [Thayer et al. 1999b]) are not testable by Y. Although some of these terms may become testable by B at some point (like $G_5 = EOO_K$, which becomes testable as soon as B gets K), they are not testable at the time B receives them, and so we represent them differently. The principal associated with a strand $s \in Resp[X, Y, T, M, K, EOR_C, EOR_K, G_2, G_3, EOO_C, EOO_K, *, *, G_1, G_4]$ is B.

5.2.2. Finish protocol

Figure 11 illustrates the ZDB finish protocol.

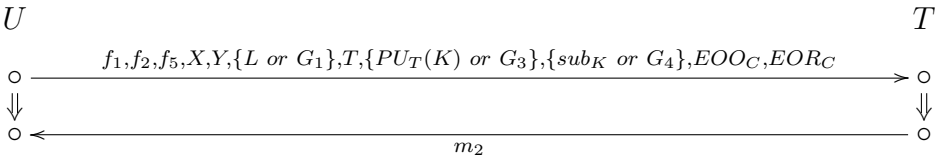

Figure 11. Finish protocol

1. A strand
$s \in Init[X, Y, T, M, K, EOR_C, EOR_K, C, PU_T(K), EOO_C, EOO_K, abort, con_K, L]$ iff s has trace of the form
$$\langle +f_1\ f_2\ f_5\ X\ Y\ L\ T\ PU_T(K)\ sub_K\ EOO_C\ EOR_C,\ -f_2 \rangle$$
where $m_2 = f_8, X, Y, L, abort$ if the protocol has been aborted and $m_2 = f_2, f_6, X, Y, L, K, con_K, EOR_C$ if the protocol has not been aborted. The principal associated with a strand $s \in Init[X, Y, T, M, K, EOR_C, EOR_K, C, PU_T(K), EOO_C, EOO_K, abort, con_K, L]$ is A.
2. A strand $s \in Resp[X, Y, T, M, K, EOR_C, EOR_K, G_2, G_3, EOO_C, EOO_K, abort, con_K, G_1, G_4]$ iff s has trace of the form
$$\langle +f_1\ f_2\ f_5\ X\ Y\ G_1\ T\ G_3\ G_4\ EOO_C\ EOR_C,\ -m_2 \rangle$$
The principal associated with a strand $s \in Resp[X, Y, T, M, K, EOR_C, EOR_K, G_2, G_3, EOO_C, EOO_K, abort, con_K, G_1, G_4]$ is B.
3. A strand $s \in Serv[X, Y, T, M, K, EOR_C, C, PU_T(K), EOO_C, abort, con_K, L, sub_K]$ iff s has trace of the form
$$\langle -f_1\ f_2\ f_5\ X\ Y\ L\ T\ PU_T(K)\ sub_K\ EOO_C\ EOR_C,\ +m_2 \rangle$$
The principal associated with a strand $s \in Serv[X, Y, T, M, K, EOR_C, C, PU_T(K), EOO_C, abort, con_K, L, sub_K]$ is T.

5.2.3. Cancel protocol

Figure 12 illustrates the ZDB cancel protocol.

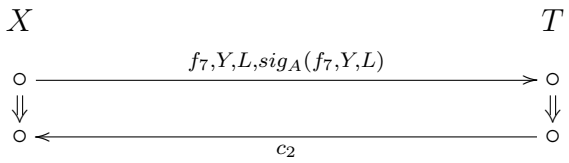

Figure 12. Cancel protocol

1. A strand $s \in Init[X, Y, T, M, K, EOR_C, EOR_K, C, PU_T(K), EOO_C, EOO_K, abort, con_K, L]$ iff s has trace of the form
$$\langle +f_7\ Y\ L\ sig_A(f_7\ Y\ L),\ -c_2 \rangle$$
where $c_2 = f_2, f_6, A, B, L, K, con_K, EOR_C$ if the protocol has already been resolved and $c_2 = f_8, A, B, L, abort$ if the protocol has not been resolved yet (by execution of the finish protocol). The principal associated with a strand $s \in Init[X, Y, T, M, K, EOR_C, EOR_K, C, PU_T(K), EOO_C, EOO_K, abort, con_K, L]$ is A.
2. A strand $s \in Serv[X, Y, T, M, K, EOR_C, C, PU_T(K), EOO_C, abort, con_K, G_1, sub_K]$ iff s has trace of the form
$$\langle -f_7\ Y\ G_1\ sig_A(f_7\ Y\ G_1),\ +c_2 \rangle$$
The principal associated with a strand $s \in Serv[X, Y, T, M, K, EOR_C, C, PU_T(K), EOO_C, abort, con_K, G_1, sub_K]$ is T.

The sets *Serv, Resp* and *Init* are pairwise disjoint.

5.3. Verification of the ZBD protocol

In this section we prove that the ZBD protocol fails to provide timeliness.

5.3.1. Timeliness

Let A be a initiator associated with a strand $s_A \in Init$. Notice that all of the parameters used to define the initiator's trace are testable by A. Because of this we can assure that every time A needs to invoke subprotocol cancel or finish, she has all the necessary terms and they are all valid. Therefore the protocol achieves timeliness for A.

Let B be a responder associated with a strand $s_B \in Resp$. Some of the parameters used to define the responder's trace can not be tested by B, and so we must proceed with the verification. As B is a responder, he can only invoke the finish subprotocol. So, applying theorem 3.5, the ZDB protocol provides timeliness for B iff all terms needed to invoke the finish protocol are testable by B.

Proof. As we can see by simple inspection, the first message of the finish protocol, when invoked by a responder, contains G_1, G_3 and G_4, which are untestable by B. When T receives these terms from B, she will test them and check if they correspond to $L = H(M, K)$, $PU_T(K)$ and $sub_K = sig_A(f_5, B, L, K, T, EOO_C)$, respectively. If any of these tests are not correct, T will judge that B has acted maliciously and will stop the finish protocol run, leaving B without any options for ending the exchange properly. We then conclude that the protocol fails to provide timeliness for B. ■

This failure makes the following attack possible: Suppose that a malicious entity A initiates an exchange with an honest principal B. She sends in the first message a bogus sub_K or a bogus $PU_T(K)$. If every other term is valid, then B will proceed and send the second message to A. Now A has everything she needs to call the cancel protocol and get the cancellation token *abort*. This constitutes a valid contract for A to end with. B, on the other hand, will invoke the finish protocol when A stops responding. He will provide every term necessary to T, but the sub_K or $PU_T(K)$ (or even both) will not be valid. Then T will quit the finish protocol, and B will remain without a valid contract. Later, in an external dispute, A is able to claim that B cheated during the exchange, as she can provide a valid contract which proves that she behaved well, while B cannot provide anything at all.

Although this attack does not result in loss of fairness for B, it leaves room for malicious behaviour that culminates on an honest principal being judged incorrectly. This attack was firstly introduced in [Gürgens et al. 2005].

6. Results and conclusions

The application of the strand spaces method to fair exchange protocols is a promising and feasible way to derive formal proofs of properties. The results confirmed the attacks discovered for the FPH protocol in [Monteiro and Dahab 2002] and for the ZDB in [Gürgens et al. 2005]. In the case of FPH, an additional attack was found, which was not previously reported. Although the proposition of solutions for the attacks is out of the scope of this work, it can be seen that this method highlights the points of failure with regard to fair exchange properties, allowing protocol analysts to devise solutions to fix them.

References

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

Boyd, C. and Kearney, P. (2000). Exploring fair exchange protocols using specification animation. In *ISW '00: Proceedings of the Third International Workshop on Information Security*, pages 209–223, London, UK. Springer-Verlag.

Chadha, R., Kanovich, M. I., and Scedrov, A. (2001). Inductive methods and contract-signing protocols. In *ACM Conference on Computer and Communications Security*, pages 176–185.

Chadha, R., Kremer, S., and Scedrov, A. (2004). Formal analysis of multi-party contract signing.

Ferrer-Gomila, J. L., Payeras-Capellà, M., and Rotger, L. (2000). An efficient protocol for certified electronic mail. In *Third International Workshop - ISW 2000*, volume 1975 of *Lecture Notes in Computer Science*, pages 237–248, Berlin. Springer-Verlag.

Garay, J. A., Jakobsson, M., and MacKenzie, P. (1999). Abuse-free optimistic contract signing. *Lecture Notes in Computer Science*, 1666:449–466.

Gürgens, S., Rudolph, C., and Vogt, H. (2005). On the security of fair non-repudiation protocols. *Int. J. Inf. Secur.*, 4(4):253–262.

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

Monteiro, J. R. and Dahab, R. (2002). An attack on a protocol for certified delivery. *Information Security - 5th International Conference, ISC 2002, Proceedings, LNCS 2433*.

Mukhamedov, A., Kremer, S., and Ritter, E. (2005). Analysis of a multi-party fair exchange protocol and formal proof of correctness in the strand space model. In Patrick, A. S. and Yung, M., editors, *Revised Papers from the 9th International Conference on Financial Cryptography and Data Security (FC'05)*, volume 3570 of *Lecture Notes in Computer Science*, pages 255–269, Roseau, The Commonwealth Of Dominica. Springer.

Thayer, F. J., Herzog, J. C., and Guttman, J. D. (1999a). Mixed strand spaces. In *CSFW '99: Proceedings of the 1999 IEEE Computer Security Foundations Workshop*, page 72, Washington, DC, USA. IEEE Computer Society.

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.

Zhou, J., Deng, R. H., and Bao, F. (1999). Evolution of fair non-repudiation with ttp. In *ACISP '99: Proceedings of the 4th Australasian Conference on Information Security and Privacy*, pages 258–269, London, UK. Springer-Verlag.

Zhou, J. and Gollmann, D. (1997). Evidence and non-repudiation. *Journal of Network and Computer Applications*, 20(3):267–281.