Analysis of a Multi-Party Fair Exchange Protocol and Formal Proof of Correctness in the Strand Space model

Steve Kremer[1], Aybek Mukhamedov[2], and Eike Ritter[2]

[1] Laboratoire Spécification et Vérification
CNRS UMR 8643 & INRIA Futurs projet SECSI & ENS Cachan, France
kremer@lsv.ens-cachan.fr
[2] School of Computer Science
University of Birmingham, UK
{A.Mukhamedov, E.Ritter}@cs.bham.ac.uk

Abstract. A multi-party fair exchange protocol is a cryptographic protocol allowing several parties to exchange commodities in such a way that everyone gives an item away if and only if it receives an item in return. In this paper we discuss a multi-party fair exchange protocol originally proposed by Franklin and Tsudik, and subsequently shown to have flaws and fixed by González and Markowitch. We identify flaws in the fixed version of the protocol, propose a corrected version, and give a formal proof of correctness in the strand space model.

1 Introduction

The problem of fairly exchanging electronic goods over a network has gained increasing importance. In a fair exchange several entities want to exchange their goods in a way that none of them will be fooled, i.e. no entity will give away his own item without also getting the other expected item. The problem arises because of an inherent asymmetry: no entity wants to be the first one to send out his item because another entity could refuse to do so after having obtained the first entity's item. In 1980, Even and Yacobi [11] showed that no deterministic contract-signing protocol—contract signing is a special case of fair exchange where digital signatures on a contract are exchanged—exists, without the participation of a trusted third party. A simple solution consists in using a trusted party (T) as an intermediary. Signers send their respective contracts to T, who first collects the contracts and then distributes them among the signers. Other solutions include randomized protocols as well as protocols based on gradual information exchange. More recently, the so-called *optimistic* approach was introduced in [3, 7]. The idea is that T intervenes only when a problem arises, i.e. some entity is trying to cheat or a network failure occurs at a crucial moment during the protocol. Such a trusted party is called *offline*. However, these protocols are less attractive when the group of entities involved in the exchange is large, because the risk of T intervening is increased.

Most protocols that have been proposed in literature are *two-party* protocols. More recently, different kinds of *multi-party fair exchange* have been considered. In [12], Franklin and Tsudik propose a classification. One can distinguish between single-unit and multi-unit exchanges. Moreover different exchange topologies are possible. While two-party non-repudiation, certified e-mail, contract signing, or more generally fair exchange protocols are very similar, in the case of a group protocol, the different topologies corresponding to particular kinds of fair exchange increase the diversity of the protocols.

Bao et al. [4] and Franklin and Tsudik [12] concentrated on a ring topology. Each entity e_i $(0 \leq i \leq n-1)$ desires an item (or a set of items) from entity $e_{i \boxminus 1}$ and offers an item (or a set of items) to entity $e_{i \boxplus 1}$, where $\boxplus$ and $\boxminus$ respectively denote addition and subtraction modulo n. Although the ring topology seems to be the simplest one for protocol design, Gonzalez and Markowitch [14] show that Franklin and Tsudik's protocol [12] is not fair and propose a new protocol.

A ring topology is not sound in non-repudiation or certified e-mail protocols: it does not make sense that one entity receives an e-mail and a distinct entity sends the corresponding receipt. The most natural generalization seems to be a star topology, i.e. a one-to-many protocol, where one entity sends a message to $n-1$ receiving entities who respond to the sender. Kremer and Markowitch [15] proposed the first multi-party non-repudiation protocols with both online and offline T. The main motivation for these protocols is a significant performance gain with respect to n two-party protocols. Afterwards, Onieva et al. [17] extended their work with online T, in order to permit the sending of different messages to each entity.

Another topology is the more general matrix topology, where each entity may desire items from a set of entities and offer items to a set of entities. Such protocols have been proposed by Asokan et al. on synchronous networks in [2]. However, asynchronous networks are more realistic.

Multi-party contract-signing protocols [1, 13, 6, 5] give raise to yet another topology. The objective is that each signer sends its signed contract on a given contract text to all other signers and that each signer receives all other signers' contract. This corresponds to a complete graph.

In the remaining of the paper we focus on ring topologies.

It is known that security protocols are error-prone and the need for applying formal methods to security protocols has been widely recognised. In this paper, we are analysing the ring fair exchange protocol proposed by Franklin and Tsudik [12] and the "corrected" version by González and Markowitch [14]. As we will see even the version of González and Markowitch contains a subtle flaw, which exploits properties such as commutativity and homomorphism. We come up with a fixed version and give a proof of correctness in the strand space model [21]. Proving this protocol correct introduces several challenges. Firstly, we are dealing with a group protocol, i.e. the number of participants is a parameter. Secondly, the protocol relies on algebraic properties which need to be abstracted properly and included in a classic Dolev-Yao like model [10] used for

our analysis. Finally, the property that we are going to prove is *fairness*, which has been studied much less than authentication or secrecy.

There have been applications of formal methods to two-party fair exchange protocols, including automated analysis using model-checkers [16, 20], as well as hand proofs in frameworks, such as CSP [19] and multi-set rewriting [8]. This list is far from being complete. Recently, Chadha et al. [9] have successfully used the model-checker MOCHA to discover errors in a multi-party contract signing protocol proposed by Garay and MacKenzie. To the best of our knowledge, this is the only formal analysis of multi-party fair exchange protocols. While a model-checker is very useful to discover errors, it does not give a proof of correctness because the analysed model needs to be considerably simplified. The advantage of a pen and paper proof is that much less simplification is required. In [18], Pereira and Quisquater used the strand space model to prove a generic insecurity result for a family of group authentication protocols. This work already demonstrates that the strand space model can be used in the case of group protocols and can be extended to model algebraic properties, e.g. Diffie-Hellman exponentiation in their case.

Our paper will be organised as follows. In the next section we present Franklin and Tsudik's protocol [12], as well as González and Markowitch's attack and fix [14]. Then we analyse this protocol, show some weaknesses and a more fundamental attack exploiting commutativity and homomorphism of the one way function used in the protocol. We go on to present how the strand space model has been adapted to our needs. Finally, we prove correctness of our fixed protocol in the strand space model and conclude.

2 Description of the protocols

In this section we describe the protocol presented by Franklin and Tsudik in [12]. We also present an attack, discovered by González and Markowitch [14] four years after the original protocol was published, as well as González and Markowitch's fixed version of the protocol. As we will see in section 3 even the fixed version of the protocol is flawed. This fact emphasizes once more the difficulty of designing correct protocols, above all group protocols, and the need for formal methods.

2.1 Notations

We use the following notations when describing the protocols:

- $\mathcal{P}$: the set of n participants;
- P_i: the ith participant of the protocol ($1 \leq i \leq n$);
- m_i: the commodity P_i wishes to exchange;
- R_i: a random number chosen by P_i.

To avoid notational clutter, we suppose that all operations on subscripts are performed modulo n. The protocol also relies on a homomorphic one-way function f, i.e. $f(x_1) \cdot f(x_2) = f(x_1 \cdot x_2)$, and a function with n arguments F_n, such

that $F_n(x_1, f(x_2), \ldots, f(x_n)) = f(x_1 \cdot x_2 \cdot \ldots \cdot x_n)$. In [12], Franklin and Tsudik propose $f(x) = x^2 \pmod N$ and $F_n(x_1, \ldots, x_n) = x_1^2 \cdot x_2 \cdot \ldots \cdot x_n$, where N is an RSA modulus.

2.2 The Franklin-Tsudik protocol

A short, informal description of the protocol for P_i is given in protocol 1. Remember that the aim of the protocol is a fair exchange on a ring topology: participant P_i has to send its commodity m_i to P_{i+1} and will receive commodity m_{i-1} from P_{i-1} in exchange.

Protocol 1 Franklin-Tsudik multi-party fair exchange protocol for P_i

1. $P_i \rightarrow P_{i+1}$: R_i
2. $P_{i-1} \rightarrow P_i$: R_{i-1}
3. $P_i \rightarrow T$: $A_i, C_i, f(R_i)$
   where $A_i = F_n(m_i, \langle f(m_k) \rangle_{k \neq i})$ and $C_i = m_i \cdot {R_i}^{-1}$
4. $T \rightarrow P_i$: $\mathcal{C}$
   where $\mathcal{C} = \{C_i \mid 1 \leq i \leq n\}$

At the end of a preliminary set-up phase it is assumed that:

- the identities of all participating parties are known;
- all participants agree on T and the functions f and F_n;
- the descriptions of the items to be exchanged, $f(m_i)$, are public.

Moreover, all channels are assumed to be private and authentic.

The protocol proceeds as follows. In a first message P_i sends a random number R_i to P_{i+1} and receives another random number R_{i-1} from P_{i-1} in the second message. Then P_i contacts the trusted party T by sending $A_i = F_n(m_i, \langle f(m_k) \rangle_{k \neq i})$, $C_i = m_i \cdot {R_i}^{-1}$ and $f(R_i)$. The trusted party T waits until it has received a message from every participant P_i. It then performs two checks:

- equality of all A_i's;
- $F_{n+1}(\prod_{1 \leq i \leq n} C_i, f(R_1), \ldots, f(R_n))$ is equal to $f(\prod_{1 \leq i \leq n} m_i)$, which should be equal to A_i.

If both checks succeed then T sends $\mathcal{C}$, the set of all C_j's, via broadcast to each P_i. Finally, P_i can check for each C_j in $\mathcal{C}$ whether $f(C_j \cdot R_{i-1}) = f(m_{i-1})$. If the check succeeds for C_j, P_i computes $m_{i-1} = C_j \cdot R_{i-1}$.

2.3 Attack and "fix" by González and Markowitch

The checks performed by T in the above described protocol are justified by the authors of the original protocol to "*establish that all m_i's and R_i's are consistent and have been properly committed*", and "*coherence of all m_i values*". However,

one may notice from the protocol that the former check does not guarantee consistency of R_i's. As a result, González and Markowitch in [14] found an attack that exploited this weakness: a dishonest participant P_i supplies different values of R_i to P_i and the trusted server T. The checks performed by T still hold, while P_{i+1} fails to receive correct multiplicative inverse from T and hence is unable to recover m_i.

González and Markowitch [14] suggest a revised protocol. They assume a preliminary setup phase which is similar to the one described above. However, they add a label ℓ to identify a protocol run, obtained by applying a one-way hash function to $\mathcal{P}$ and the set of m_i's. Knowledge of ℓ is also assumed after the setup phase. In their presentation, the authors drop the hypothesis of private and authentic channels, but sign and encrypt each of the messages explicitly. In our presentation here, for the sake of simplicity, we assume private and authentic channels, as signatures and encryption is only one possible way of achieving these goals. A short description of the protocol is given in protocol 2. The main difference with the previous protocol is that P_i includes $f(R_{i-1})$ in his message when contacting T. This change should prevent the previous attack. The authors give an informal argument of its correctness. Unfortunately, as we will show, not all attacks are avoided.

Protocol 2 González-Markowitch multi-party fair exchange protocol for P_i

1. $P_i \rightarrow P_{i+1}$: ℓ, R_i
2. $P_{i-1} \rightarrow P_i$: ℓ, R_{i-1}
3. $P_i \rightarrow T$: $\ell, A_i, C_i, f(R_i), f(R_{i-1})$
   where $A_i = F_n(m_i, \langle f(m_k) \rangle_{k \neq i})$ and $C_i = m_i \cdot {R_i}^{-1}$
4. $T \rightarrow P_i$: $\ell, \mathcal{C}$
   where $\mathcal{C} = \{C_i \mid 1 \leq i \leq n\}$

3 Analysis

3.1 Implicit assumptions

In both papers [12] and [14], the properties of commodities exchanged are not specified and ambiguous. As a result, no (explicit) restrictions are put on dishonest agents against copying and distributing them. In particular, an honest agent P is left in an unfair state if after the first cycle a dishonest agent $\widetilde{P}$, who received P's item, decides to distribute this item to others.

The above described weakness demonstrates that it is impossible to guarantee fairness for an exchange of arbitrary items. Therefore, we explicitly need to make the following assumptions about the properties of commodities:

- commodities are token-based, i.e. an agent loses ownership of m_i when he gives it away and there is at most one owner of m_i at any time;

– or, a commodity can only be issued by an originator and used by the party it is issued to.

These properties are outside the scope of the protocol and need to be ensured by other techniques. However, we believe that they need to be explicitly stated.

3.2 Replay Attacks

In [14], the label ℓ was introduced to serve as an identifier of a protocol session. However, the resulting label is not unique, which allows the following attack.

Suppose P_i, P_k and $\widetilde{P}$ decide to perform a cyclic exchange, where P_i sends an item to P_k in return for an item from $\widetilde{P}$. After the setup phase, $\widetilde{P}$ observes all messages sent by P_i. As protocol messages are assumed to be private and authentic, the intruder can neither elicit their components nor claim to an other party to be their originator; he also can't replay the intercepted message containing a nonce to other honest parties, as it contains the recipient's identity. In this run $\widetilde{P}$ may or may not send a message to T, who eventually stops the protocol after some pre-defined amount of time if the latter is chosen.

Suppose that the same cyclic exchange takes place after $\widetilde{P}$ retrieves a nonce R_i. The "label" corresponding to this run will be the same as in the previous one. Channels are assumed to be resilient [14], which means that messages can get delayed by an arbitrary but finite amount of time. Therefore, after the setup phase the intruder can delay the nonce ${R_i}'$ from P_i intended to P_k, long enough such that: (i) he can replay the message containing R_i from the previous run to P_k; (ii) P_k sends his message to T. $\widetilde{P}$ also replays the other messages of P_i to T, as well as his previous message to T (except $f(R'_k)$ substituted for $f(R_k)$), but sends the unmatching nonce to P_i afterwards. All checks succeed and T broadcasts $\mathcal{C}$. P_k gets m_i and $\widetilde{P}$ gets m_k and m_i; P_i does not get $\widetilde{m}$ and even if she acquires $\widetilde{P}$'s message, she is still in an unfair state as the only expected recipient of her message is supposed to be P_k.

In a simpler version of the above attack dishonest agents simply send new nonces in a replay of the protocol to leave regular agents in an unfair state.

To conclude, in the analysis of replay weaknesses we need to make a stronger assumption on the setup phase: an intruder cannot simply via replaying messages of the setup phase make T to vacuously believe that another exchange is initiated, and all participants of the exchange know true identities of the others, as otherwise, replay attacks are trivial.

3.3 Arithmetic attack

We now present a more fundamental and interesting attack. The protocol presented in [14] (protocol 2) ensures consistency between the nonces sent by P_i to P_{i+1} with respect to T. However, it does not address consistency among C_i and $f(R_i)$: it is not ensured that the former contains a multiplicative inverse of R_i. The second check performed by T,

$$F_{n+1}(\prod_{1\leq i\leq n} C_i, f(R_1), \ldots, f(R_n)) \stackrel{?}{=} f(\prod_{1\leq i\leq n} m_i)$$

only verifies if an agent has supplied m_i in C_i which is consistent with expectations of other agents[1]. As a result, fairness can be broken by two non-contiguous[2] cooperating dishonest agents, who receive desired exchange messages while not revealing theirs. The following details the attack.

The protocol proceeds as described in [14], except that two dishonest non-contiguous agents $\widetilde{P}_i$ and $\widetilde{P}_k$ send $C_i = m_i \cdot {R_k}^{-1}$ and $C_k = m_k \cdot {R_i}^{-1}$ to T, respectively. All the checks of T will succeed, including

$$\begin{aligned} & F_{n+1}(\textstyle\prod_{1\leq i\leq n} C_i, f(R_1), \ldots, f(R_n)) \\ = {} & F_{n+1}(m_1 R_1^{-1} \cdot \ldots \cdot m_i R_k^{-1} \cdot \ldots \cdot m_k R_i^{-1} \cdot \ldots m_n R_n^{-1}, f(R_1), \ldots, f(R_n)) \\ = {} & f(\textstyle\prod_{1\leq i\leq n} m_i). \end{aligned}$$

However, honest parties P_{i+1} and P_{k+1} will not receive correct multiplicative inverses. Thus, they are not able to recover m_i and m_k, respectively. This attack is weak in the sense that P_{i+1} could recover m_k, i.e. a secret of $\widetilde{P}_i$'s conspirator. However, P_{i+1} would need to test all possible item descriptions which is not foreseen in the protocol. Moreover, there exists a stronger attack, where this is not possible: $\widetilde{P}_i$ could send $C_i = m_i \cdot m_k$ and $\widetilde{P}_k$ would send $C_k = R_i^{-1} \cdot R_k^{-1}$.

This attack was first discovered when trying to prove correctness of the González-Markowitch protocol in the strand space model. The failure of the proof hinted directly towards the above attack and illustrates that strand spaces are also useful to discover new flaws.

4 Formal model

We use the well-studied strand space model [21] to represent and prove correct a fixed version of the above analysed protocol. Basic definitions and facts about strand spaces are recalled in Appendix A. In this section we only provide intuitions about strand spaces without giving the formal definitions.

We start defining the set of possible messages. We need to extend the sets of messages classically used in strand spaces in order to deal with algebraic properties, such as the inverses, used in the protocols presented before. This way of abstracting algebraic properties over groups is inspired by [18], where the authors handle Diffie-Hellman exponentiations.

Definition 1. *Let:*

1. T *be the set of texts;*
2. R *be the set of random values used in the protocol;*
3. M *be the set of commodities exchanged in the protocol;*
4. C *be the set of ciphertexts;*

[1] Due to associativity and commutativity of multiplication it does not ensure any further consistency.

[2] Otherwise, one of them will not receive a secret.

5. $(\mathbf{G_{rm}}, \cdot)$ *be the commutative group freely generated from elements in* R *and* M*; the unit element is denoted* 1*;* $\underbrace{g \cdot \ldots \cdot g}_{n\times}$ *is denoted* g^n *and* $g^0 = 1$*. Similarly,* C *freely generates group* $(\mathbf{G_c}, \cdot)$ *that is also abelian. Let* $\mathbf{G} = \mathbf{G_{rm}} \cup \mathbf{G_c}$*;*
6. $\mathsf{inv} : \mathbf{G} \to \mathbf{G}$ *be an injective function computing the inverse element, such that* $\mathsf{inv}(x) \cdot x = 1$*.* $inv(x)$ *is also denoted* x^{-1}*;*
7. $\mathsf{f} : \mathbf{G_{rm}} \to \mathbf{G_c}$ *be a homomorphic one-way function;*
8. $(\mathsf{A}, \cdot)$ *be the free monoid generated by* $\mathbf{G}$*. It represents the free term algebra of our protocol;*
9. A' *be the set of elements disjoint from the message algebra of the protocol and define a map* $|\cdot| : \mathsf{A} \to \mathsf{A}'$*, such that* $|m| = |n|$ *iff* $m = n$*. This is an auxiliary construct to allow abstraction in defining private and authentic channels;*
10. $\mathsf{A} \cup \mathsf{A}'$ *is the set of all terms that can appear in our model of the protocol.*

We now introduce the subterm relation $\sqsubseteq$.

Definition 2. *Let* a *be an atom. Then* $\sqsubseteq$ *is the smallest inductively defined relation such that*

- $a \sqsubseteq a$*;*
- $a \sqsubseteq gh$ *if* $a \sqsubseteq g$ *or* $a \sqsubseteq h$*;*
- $a \sqsubseteq g \in \mathbf{G}$ *if* $g = g_1^{k_1} \cdot \ldots \cdot g_n^{k_n}$*,* g_i*'s are atoms of* $\mathbf{G_{rm}}$ *or* $\mathbf{G_c}$*,* $i \neq j \Rightarrow g_i \neq g_j$ *and* $\exists \ell (a = g_\ell^{k_\ell} \wedge k_\ell \neq 0)$ $(1 \leq i, j, \ell \leq n)$*;*
- $a \sqsubseteq \mathsf{f}(x)$ *if* $a \sqsubseteq x$*;*
- $a \sqsubseteq |m|$ *if* $a \sqsubseteq m$*.*

For example, we have that $R_1 \sqsubseteq \mathsf{f}(R_1 \cdot R_2)$ and $R_1 \not\sqsubseteq R_2 \cdot R_3$ (even though $R_2 \cdot R_3 = R_2 \cdot R_3 \cdot R_1 \cdot R_1^{-1}$), where R_1, R_2 and $R_3 \in \mathsf{R}$.

Informally, a *strand* is a sequence of message transmissions and receptions, representing a role of the protocol. Transmission of the term t is denoted $+t$, while reception of t is written $-t$. Each transmission, respectively reception, corresponds to a *node* of the strand and we denote the ith node of strand s as $\langle s, i \rangle$. The relation $n \Longrightarrow n'$ holds between nodes n and n' on the strand s if $n = \langle s, i \rangle$ and $n' = \langle s, i + 1 \rangle$. The relation $n \longrightarrow n'$ represents communication between strands and holds between nodes n and n' if $term(n) = +t$ and $term(n') = -t$ ($term(n)$ denotes the signed term corresponding to node n, $unsterm(n)$ denotes the unsigned term corresponding to node n). A *strand space* Σ is a set of strands and the relations $\Longrightarrow$ and $\longrightarrow$ impose a graph structure on the nodes of Σ. A *bundle* is a subgraph of the graph defined by the nodes of Σ and $\Longrightarrow \cup \longrightarrow$. It represents a possible execution of the protocol. One possible bundle represents the expected protocol execution. The problem when analyzing a security protocol is that an intruder can insert or manipulate messages and may give raise to bundles not foreseen by the protocol designers.

Next we specify the possible actions of the intruder. The list of possible actions builds in some assumptions about the cryptographic system used. Firstly, we assume that it is impossible for the intruder to do factorisation, in other words

given a product $g \cdot h$, the intruder cannot deduce g or h from them. Secondly, the intruder has no operation available which allows it to deduce any non-guessable element from any other element in G. Thirdly, the intruder cannot invert hash function f: there is no way to deduce g from $\mathsf{f}(g)$.

We have the following definition of intruder traces:

Definition 3. *An* intruder trace *is one of the following:*

- ***M****. Text message:* $\langle +t \rangle$*, where* t *is a guessable element of* A *(i.e. in* $T \cup C$*);*
- ***F****. Flushing:* $\langle -g \rangle$*;*
- ***T****. Tee:* $\langle -g, +g, +g \rangle$*;*
- ***C****. String concatenation:* $\langle -g, -h, +gh \rangle$*;*
- ***S****. String decomposition:* $\langle -gh, +g + h \rangle$*;*
- ***Op****. Arithmetic operation:* $\langle -g, -h, +(g \cdot h) \rangle$*, where the binary operator can be either from* $\mathbf{G_{rm}}$ *or* $\mathbf{G_c}$*.*
- ***Apf****. Application of a hash function* f*:* $\langle -g, +\mathsf{f}(g) \rangle$*.*

We define private, authenticated and resilient channels used in the protocol:

Definition 4. *Suppose* A *sends* B *message* m*. Then for some strand* $s_A \in A[*]$ $\exists i.term(\langle s_A, i \rangle) = m$*. Let* $\mathcal{C}$ *be any bundle where* $s_A \in \mathcal{C}$*, and consider the set* $S = \{n \in \mathcal{C} | \langle s_A, i \rangle \prec n \wedge k \sqsubseteq term(n)\}$[3] *for some non-guessable* $k \sqsubseteq m$*. We say there is a* private channel *between* A *and* B *if* $\forall n \in S.\ |m| \not\sqsubseteq term(n)$ *implies:*

- *either* $\exists i. \langle s_B, i \rangle \in S$ *and* $\langle s_B, i \rangle \preceq n$*;*
- *or there is* $\prec$*-minimal element* n'' *of the set* $\{n' \in \mathcal{C} | n' \prec n \wedge k \sqsubseteq term(n')\}$*, such that* $\langle s_A, i \rangle \not\prec n''$*.*

Intuitively, in our definition $|m|$ corresponds to an "encryption" of m that the intruder can only duplicate or intercept. Moreover, a non-guessable m (or part of it) can never be derived while transmitted over a private channel, unless the secret was revealed where the sender in the private channel did not causally contribute to the compromise.

Definition 5. *Suppose* B *receives a message* m *apparently from* A *- for some strand* $s_B \in B[*]$ $\exists i.term(\langle s_B, i \rangle) = m$*. Let* $\mathcal{C}$ *be any bundle where* $s_B \in \mathcal{C}$*, and consider the set* $S = \{n \in \mathcal{C} | n \prec \langle s_B, i \rangle \wedge m \sqsubseteq term(n)\}$*. We say there is an* authenticated channel *between* A *and* B *if* $\exists n \in S.\ strand(n) = s_A$*.*[4]

A private (or authenticated) channel for a protocol Π can be implemented by any protocol that guarantees secrecy (or authetication) in a composition with Π (which may also subsume other primitive protocols).

We use the notion of a *reference bundle* in the next definition. Intuitively, it represents the full execution of the protocol without intruder. In the language of

[3] Logical connectives have the largest scope.

[4] Disregarding m this definition corresponds to aliveness - the weakest type of authentication of Lowe's hierarchy. It could be lifted up to non-injective agreement by choosing an appropriate m.

strands, a bundle is a reference bundle ($\mathcal{C}_r$) if for any regular role A of a protocol $\exists! s_A \in \mathcal{C}_r$, such that $length(tr(s_A)) = max(\{length(tr(s))|s \in A[*]\})$, and no intruder strand is in $\mathcal{C}_r$.

Definition 6. *Suppose A sends B message m. Then for some strand* $s_A \in A[*]$ $\exists i.term(\langle s_A, i\rangle) = m$ *and let* $\mathcal{C}$ *be any bundle where* $s_A \in \mathcal{C}$*. We say there is a* resilient channel *between A and B if* $\{R \in roles\ |\exists s_R \in \mathcal{C} \wedge \exists k.term(\langle s_R, k\rangle) = -m \wedge \langle s_A, i\rangle \prec \langle s_R, k\rangle\} = \{R \in roles\ |\exists s'_R \in \mathcal{C}_r \wedge \exists k.\langle s_A, i\rangle \longrightarrow \langle s'_R, k\rangle\}$.

We are now able to give a description of the protocol in strand spaces.

5 Fixing the protocol and proof of correctness

5.1 Fixing the Protocol

Replay attacks exploit possible collision in the label space. An easy fix is to make T distribute a fresh (possibly guessable) value to participants of the exchange at setup phase, which needs to be included in all messages in that run of the protocol[5]. However, to guarantee uniqueness in strand spaces T's name needs to be associated with it, viz. we form a tuple (T, n) and call it a *tag*. Obviously, for a tag to work, we need to disallow branching on T's strand, viz. for any run of a setup phase, T executes at most one run of the protocol. Furthermore, if n in a tag is not a timestamp, *identical* setup phase requests must be distinguished, where timestamps or "handshake" mechanism have to be used. Namely, our assumption on a setup phase is: if bundles B_1 and B_2 represent *identical* setup phase runs, i.e. with the same participants and messages to be exchanged, occurring at times t_1 and t_2, respectively, then T-strands have different tags.

An intuitive fix to the arithmetic attack on the protocol is via making T to verify consistency of multiplicative inverses, e.g. instead of the second check performed by T in the González-Markowitch protocol, we suggest that T verifies that

$$\forall i.\exists C \in \mathbf{C} \text{ such that } f(C \cdot R_{i-1}) = f(m_{i-1})$$

where m_{i-1} is the item required by P_i. In order to perform this check T needs to know agent-message correspondence, which can be established either at setup phase, or by P_i sending $f(m_{i-1})$ in the message to T. Note that the second protocol proposed in [12], which we did not study here, uses similar tests to ensure consistency.

5.2 A strand space model of our corrected protocol

We now specify our corrected version of the protocol using the strand space formalism. The parameterized P_i strand representing any honest participant P_i is as follows:

$$P_i[t, m_i, \mathtt{f}(m_1), \ldots, \mathtt{f}(m_n), R_i] = \langle +t\ R_i, -t\ R_{i-1}, +t\ A_i\ C_i\ \mathtt{f}(R_i), -t\ \mathbf{C}\rangle$$

[5] Thus, timestamps or nonces may suffice for this purpose.

where $t \in \mathsf{T}$, $m_i \in \mathsf{M}$, $R_i, R_{i-1} \in \mathsf{R}$, $A_i = \mathsf{F}_n(m_i, \langle \mathsf{f}(m_k) \rangle_{k \neq i})$, $C_i = m_i \cdot {R_i}^{-1}$ and $\mathcal{C}$ is any concatenation of C_js ($1 \leq i \leq n$).

The role of the trusted party corresponds to the following strand:

$$T[t, \mathsf{f}(m_1), \ldots, \mathsf{f}(m_n)] = \langle -t\ A_1\ C_1\ \mathsf{f}(R_1), \ldots, -t\ A_n\ C_n\ \mathsf{f}(R_n), +t\ \mathbf{C} \rangle$$

The parametric P_i and T strands represent set of strands, i.e. the union of all possible instantiations. We denote these sets by $P_i[*]$ and $T[*]$ respectively.

Moreover, we make the following assumptions:

- all channels are private, authentic and resilient;
- during a setup phase all participants agree on a trusted party T who generates a unique tag t, and at most one strand of each regular role receives it;
- $a = b$ iff $\mathsf{f}(a) = \mathsf{f}(b)$.

5.3 Correctness in Strands Model

Fairness Fairness is the central property of our protocol. In both papers [12] and [14], *fairness* is defined as the property, whereby an honest participant receives all expected items corresponding to the items he has provided. This formulation is not formal enough for our analysis and, hence, the following definition is adopted:

Definition 7. *A multi-party cyclic fair exchange protocol is* fair *for a honest agent P_i if P_{i+1} obtains m_i only if P_i obtains m_{i-1}.*

Definition 8. *The protocol space is a collection of the following types of strands:*

1. *honest agent strands $s_i \in P_i[*]$;*
2. *trusted party strands $t \in T[*]$;*
3. *the penetrator strands, modeling dishonest agents.*

Proof of fairness

Proposition 1. *The protocol guarantees fairness.*

Lemma 1. *A fresh value n generated by T in a setup phase uniquely identifies the strand of T in the protocol and a tag (T, n) uniquely identifies regular strands of the protocol.*

Proof. Trivial from assumptions.

Lemma 2. *Suppose a collection of agents $\mathcal{P}$ runs the protocol, where $P_i \in \mathcal{P}$ is an honest agent, T is the trusted party and (T, n) is a tag. Then no agent $P_j \in \mathcal{P}$ can obtain m_i without obtaining $\mathbf{C}$, where $\mathbf{C} \sqsubseteq unsterm(\langle \mathbf{t}, \mathbf{1} \rangle)$, $(T, n) \sqsubseteq unsterm(\langle t, 1 \rangle)$ and $t \in T[*]$.*

Proof. By assumption on a setup phase, all commodities to be exchanged are secret before the execution of the protocol. Let $s_i \in P_i[*]$ in some bundle $\mathcal{C}$. We have that $m_i \not\sqsubseteq \langle s_i, 0\rangle$ and $m_i \not\sqsubseteq \langle s_i, 1\rangle$. Hence, the two first messages do not reveal m_i and the only event that may reveal m_i is a node $nd = \langle s_i, 2\rangle$. By the assumption of resilient channels, $\exists t_T \in \mathcal{C}$, such that $\exists i.term(\langle t_T, i\rangle) = term(nd)$ and the node $nd' = \langle t_T, i\rangle$ is negative.

Consider the message $m = term(nd)$. $m_i \sqsubseteq m$ and m_i uniquely originates on nd. Hence, no node $n \in \mathcal{C}$ exists, such that $m_i \sqsubseteq term(n)$ and $nd \not\prec n$. Let $S = \{n \in \mathcal{C} | nd \prec n \wedge m_i \sqsubseteq term(n)\}$. So, by the assumsion of private channels, we have $\forall n \in S. |m_i| \sqsubseteq term(n)$, unless $nd' \prec n$. In other words, m_i is uncompromised up until T receives it. Assuming that P_i and T remain honest in other runs of the protocol, as $(T, n) \sqsubseteq m$, by Lemma 1 replay of the message in any other run of the protocol will be rejected. Lastly, $length(t) = 2$ and $\mathbf{C} \sqsubseteq unsterm(\langle t, 1\rangle)$. Hence, the lemma holds. □

Corollary 1. *Fairness is preserved up to and including the third event on each honest agent's strand.*

Lemma 3. *For every honest P_i, P_{i+1} obtains m_i only if P_i obtains m_{i-1}.*

Proof. Consider an honest agent P_i, participating in a cyclic exchange. Assume that P_i completed all but the last steps of the protocol. There are two cases to consider:

1. P_i does not receive the last message containing $\mathbf{C}$.
   As the agent-server link is assumed to be resilient, T did not send $\mathbf{C}$ to P_i[6]. By assumption on a setup phase, T is informed of all participants in the exchange and, as a result, T did not send $\mathbf{C}$: either T did not receive all expected messages or one of the checks did not succeed. In any case, T did not send $\mathbf{C}$ to any other agent, and by Lemma 2 m_i cannot be obtained by any other agent.
2. P_i does receive the last message containing $\mathbf{C}$
   We need to show that $\exists C_{i-1} \in \mathbf{C}$, such that $C_{i-1} \cdot R_{i-1} = m_{i-1}$, where R_{i-1} is sent by P_{i-1} to P_i. Assume the opposite, that $\forall C \in \mathbf{C}.\ C \cdot R_{i-1} \neq m_{i-1}$. According to our fix, T checks if $\exists C \in \mathbf{C}$ such that $f(C \cdot R) = f(m_i)$, where m_i is the item required by P_i. $R = R_{i-1}$ as P_i sends $f(R_{i-1})$ to T. So, T's check also fails and it does not transmit $\mathbf{C}$ – a contradiction. Therefore, $\exists C \in \mathbf{C}$, s.t. $C \cdot R_{i-1} = m_{i-1}$. This means that if T transmits $\mathbf{C}$ then $\forall i.P_i \in \mathcal{P}$ gets m_i.
   By Lemma 2 and the above argument the current lemma holds.

□

The last lemma proves our proposition. Indeed, it shows that the protocol is *(n-1)-resilient*, viz. even if all other agents are dishonest, fairness is guaranteed to the honest participant.

[6] Such statements can be made strictly formal by routine unwinding of defintions we gave previously, as it was done in the previous lemma.

6 Conclusion

In this paper we analysed a multi-party ring fair exchange protocol. The protocol has first been analysed by González and Markowitch, who discovered an attack and proposed a fix. The correctness of their fix was discussed using informal arguments. We show that their protocol is still flawed. Using the strand space model, which we adapted to model properties such as homomorphism and commutativity, we prove correctness of a modified version. The paper demonstrates again the difficulty of designing correct protocols, above all group protocols, and the crucial need for including formal methods for design and validation.

References

1. N. Asokan, Birgit Baum-Waidner, Matthias Schunter, and Michael Waidner. Optimistic synchronous multi-party contract signing. Research Report RZ 3089, IBM Research Division, December 1998.
2. N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for multi-party fair exchange. Research Report RZ 2892 (# 90840), IBM Research, December 1996.
3. N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *4th ACM Conference on Computer and Communications Security*, Zurich, Switzerland, April 1997. ACM Press.
4. Feng Bao, Robert H. Deng, Khanh Quoc Nguyen, and Vijay Varadharajan. Multi-party fair exchange with an off-line trusted neutral party. In *DEXA 1999 Workshop on Electronic Commerce and Security*, Florence, Italy, September 1999.
5. Birgit Baum-Waidner. Optimistic asynchronous multi-party contract signing with reduced number of rounds. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, ICALP 2001*, volume 2076 of *Lecture Notes in Computer Science*, pages 898–911, Crete, Greece, July 2001. Springer-Verlag.
6. Birgit Baum-Waidner and Michael Waidner. Round-optimal and abuse free optimistic multi-party contract signing. In *Automata, Languages and Programming — ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 524–535, Geneva, Switzerland, July 2000. Springer-Verlag.
7. Holger Bürk and Andreas Pfitzmann. Value exchange systems enabling security and unobservability. In *Computers and Security, 9(8):715–721*, 1990.
8. Rohit Chadha, Max Kanovich, and Andre Scedrov. Inductive methods and contract-signing protocols. In *8th ACM Conference on Computer and Communications Security*, Philadelphia, PA, USA, November 2001. ACM Press.
9. Rohit Chadha, Steve Kremer, and Andre Scedrov. Formal analysis of multi-party fair exchange protocols. In Riccardo Focardi, editor, *17th IEEE Computer Security Foundations Workshop*, pages 266–279, Asilomar, CA, USA, June 2004. IEEE Computer Society Press.
10. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
11. Shimon Even and Yacov Yacobi. Relations among public key signature systems. Technical Report 175, Technion, Haifa, Israel, March 1980.

12. Matthew K. Franklin and Gene Tsudik. Secure group barter: Multi-party fair exchange with semi-trusted neutral parties. In Ray Hirschfeld, editor, *Second Conference on Financial Cryptography (FC 1998)*, volume 1465 of *Lecture Notes in Computer Science*, pages 90–102, Anguilla, British West Indies, February 1998. International Financial Cryptography Association (IFCA), Springer-Verlag.
13. Juan A. Garay and Philip D. MacKenzie. Abuse-free multi-party contract signing. In *International Symposium on Distributed Computing*, volume 1693 of *Lecture Notes in Computer Science*, Bratislava, Slavak Republic, September 1999. Springer-Verlag.
14. Nicolás González-Deleito and Olivier Markowitch. Exclusion-freeness in multi-party exchange protocols. In *5th Information Security Conference*, volume 2433 of *Lecture Notes in Computer Science*, pages 200–209. Springer-Verlag, September 2002.
15. Steve Kremer and Olivier Markowitch. Fair multi-party non-repudiation. *International Journal on Information Security*, 1(4):223–235, July 2003.
16. Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. In Kim G. Larsen and Mogens Nielsen, editors, *Concurrency Theory—CONCUR 2001*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565, Aalborg, Denmark, August 2001. Springer-Verlag.
17. Jose Onieva, Jianying Zhou, Mildrey Carbonell, and Javier Lopez. A multi-party non-repudiation protocol for exchange of different messages. In *18th IFIP International Information Security Conference*, Athens, Greece, May 2003. Kluwer.
18. Olivier Pereira and Jean-Jacques Quisquater. Generic insecurity of cliques-type authenticated group key agreement protocols. In Riccardo Focardi, editor, *17th IEEE Computer Security Foundations Workshop*, pages 16–29, Asilomar, CA, USA, June 2004. IEEE Computer Society Press.
19. Steve A. Schneider. Formal analysis of a non-repudiation protocol. In *11th IEEE Computer Security Foundations Workshop*, pages 54–65, Washington - Brussels - Tokyo, June 1998. IEEE.
20. Vitaly Shmatikov and John Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science, special issue on Theoretical Foundations of Security Analysis and Design*, 283(2):419–450, 2002.
21. F. Javier Thayer Fabrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

A Strand spaces

We here give basic definitions about strand spaces and bundles taken from [21].

Definition 9. *Let* A *be the set of all terms. A signed term is a pair* $\langle \sigma, a \rangle$ *with* $a \in \mathsf{A}$ *and* σ *one of the symbols* $+, -$*. We will write a signed term as* $+t$ *or* $-t$*.* $(\pm\mathsf{A})$ *is the set of finite sequences of signed terms. We will denote a typical element of* $(\pm\mathsf{A})$ *by* $\langle\langle \sigma_1, a_1 \rangle, \ldots, \langle \sigma_n, a_n \rangle\rangle$*.*

A strand space over A *is a set* Σ *with a trace mapping* $tr : \Sigma \to (\pm A)^*$*.*

By abuse of language, we will still treat signed terms as ordinary terms. For instance, we shall refer to subterms of signed terms. We will usually represent a strand space by its underlying set of strands.

Definition 10. *Fix a strand space Σ.*

1. *A* node *is a pair $\langle s, i\rangle$, with $s \in \Sigma$ and i an integer satisfying $1 \leq i \leq length(tr(s))$. The set of nodes is denoted by $\mathcal{N}$. We will say the node $\langle s, i\rangle$ belongs to the strand s. Clearly, every node belongs to a unique strand.*
2. *If $\langle s, i\rangle \in \mathcal{N}$ then $index(n) = i$ and $strand(n) = s$. Define $term(n)$ to be $(tr(s))_i$, i.e. the ith signed term in the trace of s. Similarly, $unsterm(n)$ is $((tr(s))_i)_2$, i.e. the unsigned part of the ith signed term in the trace of s.*
3. *There is an edge $n_1 \longrightarrow n_2$ if and only if $term(n_1) = +a$ and $term(n_2) = -a$ for some $a \in \mathsf{A}$. Intuitively, the edge means that node n_1 sends the message a, which is received by n_2, recording a potential causal link between those strands.*
4. *When $n_1 = \langle s, i\rangle$ and $n_2 = \langle s, i+1\rangle$ are members of $\mathcal{N}$, there is an edge $n_1 \Longrightarrow n_2$. Intuitively, the edge expresses that n_1 is an immediate causal predecessor of n_2 on the strand s. We write $n' \Longrightarrow^+ n$ to mean that n' precedes n (not necessarily immediately) on the same strand.*
5. *An unsigned term t* occurs in *$n \in \mathcal{N}$ iff $t \sqsubseteq term(n)$.*
6. *Suppose I is a set of unsigned terms. The node $n \in \mathcal{N}$ is an entry point for I iff $term(n) = +t$ for some $t \in I$, and whenever $n' \Longrightarrow^+ n$, $term(n') \notin I$.*
7. *An unsigned term t* originates *on $n \in \mathcal{N}$ iff n is an entry point for the set $I = \{t' \mid t \sqsubseteq t'\}$.*
8. *An unsigned term t is* uniquely originating *in a set of nodes $S \subset \mathcal{N}$ iff there is a unique $n \in S$ such that t originates on n.*
9. *An unsigned term t is non-originating in a set of nodes $S \subset \mathcal{N}$ iff there is no $n \in S$ such that t originates on n.*

If a term t originates uniquely in a suitable set of nodes, then it can play the role of a nonce or session key, assuming that everything that the penetrator does in some scenario is in that set of nodes.

A parameterized strand, also called a *role*, is a strand which contains variables. The regular strands are generated by filling in the parameters with appropriate values. We write $s \in A[*]$ or, simply s_A, to mean that strand s corresponds to a role A. By *roles* we mean the set of all regular participants of the protocol in consideration.

$\mathcal{N}$ together with both sets of edges $n_1 \longrightarrow n_2$ and $n_1 \Longrightarrow n_2$ is a directed graph $\langle \mathcal{N}, (\longrightarrow \cup \Longrightarrow)\rangle$.

A *bundle* is a finite subgraph of $\langle \mathcal{N}, (\longrightarrow \cup \Longrightarrow)\rangle$, for which we can regard the edges as expressing the causal dependencies of the nodes, Causal dependence is expressed by $\prec = (\longrightarrow \cup \Longrightarrow)^+$ and $\preceq$ is the reflexive version of $\prec$.

Definition 11. *Suppose $\longrightarrow_{\mathcal{B}} \subset \longrightarrow$, $\Longrightarrow_{\mathcal{B}} \subset \Longrightarrow$ and $\mathcal{B} = \langle \mathcal{N}_{\mathcal{B}}, (\longrightarrow_{\mathcal{B}} \cup \Longrightarrow_{\mathcal{B}})\rangle$ is a subgraph of $\langle \mathcal{N}, (\longrightarrow \cup \Longrightarrow)\rangle$. $\mathcal{B}$ is a bundle if:*

1. *$\mathcal{N}_{\mathcal{B}}$ and $\longrightarrow_{\mathcal{B}} \cup \Longrightarrow_{\mathcal{B}}$ are finite.*
2. *If $n_2 \in \mathcal{N}_{\mathcal{B}}$ and $term(n_2)$ is negative, then there is a unique n_1 such that $n_1 \longrightarrow_{\mathcal{B}} n_2$.*
3. *If $n_2 \in \mathcal{N}_{\mathcal{B}}$ and $n_1 \Longrightarrow n_2$ then $n_1 \Longrightarrow_{\mathcal{B}} n_2$.*
4. *$\mathcal{B}$ is acyclic.*

By abuse of notation we write $n \in \mathcal{B}$ to mean $n \in \mathcal{N}_{\mathcal{B}}$.