Minimal message complexity of asynchronous multi-party contract signing

Sjouke Mauw
University of Luxembourg
sjouke.mauw@uni.lu

Saša Radomirović
University of Luxembourg
sasa.radomirovic@uni.lu

Mohammad Torabi Dashti
ETH Zürich
torabidm@inf.ethz.ch

Abstract

Multi-party contract signing protocols specify how a number of signers can cooperate in achieving a fully signed contract, even in the presence of dishonest signers. This problem has been studied in different settings, yielding solutions of varying complexity. Here we assume presence of a trusted third party that will be contacted only in case of a conflict, asynchronous communication, and a totally ordered protocol. Our goal is to develop a lower bound on the number of messages in such a protocol. Using the notion of abort chaining, which is a specific type of attack on the fairness of signing protocols, we derive the lower bound $\alpha^2 + 1$, with $\alpha > 2$ being the number of involved signers. In order to achieve this lower bound, we relate the problem of developing fair signing protocols to the open combinatorial problem of finding shortest permutation sequences. This relation also indicates a way to construct signing protocols which are shorter than current state-of-the-art protocols. We illustrate this by presenting the shortest three-party fair contract signing protocol.

1. Introduction

Contract signing protocols deal with the situation where α parties wish to sign a publicly-known contract text c, in a *fair* manner. Informally, fairness means that either every honest party obtains the signature of all the other $\alpha - 1$ parties on c, or none of the parties obtains the signature of any honest party on c; this definition is made precise in the following sections.

Deterministic fair contract signing protocols cannot be constructed in asynchronous systems with no presumed trust [14]. A way to circumvent this impossibility is to resort to randomized protocols (see [15] for a survey). Randomized protocols nonetheless require exchanging a large number of messages to approximate fairness, and usually assume that the involved parties have nearly equal computational powers. A second way to construct fair contract signing protocols is to add an impartial process to the system that is trusted by all the protocol participants; the impartial process is thus called a trusted third party (TTP). In the *optimistic* family of TTP-based contract signing protocols, the participants execute a main (or, optimistic) sub-protocol which does not involve the TTP at all. However, should a failure occur maliciously or accidentally, the participants are provided with fall-back scenarios, which enable them to recover to a fair state with the TTP's help.

The premise of optimistic protocols is that failures are infrequent, thus fall-back sub-protocols only need to be executed rarely. Therefore, a meaningful measure of efficiency in these protocols is the number of messages exchanged in the main protocol. Pfitzmann, Schunter, and Waidner have shown that in two-party contract signing protocols, four messages in the main protocol are necessary and sufficient to achieve fairness [23]. We study in this paper the problem of determining the number of messages necessary to achieve fairness in multi-party contract signing protocols.

Our approach is based on the notion of *abort chaining*, as introduced by Mukhamedov and Ryan [21]. An abort-chaining attack on an optimistic contract signing protocol consists of a sequence of requests by the signers to the TTP, forcing the TTP to abort the protocol, even though one of the honest signers already signed the contract. Mukhamedov and Ryan used abort chaining to prove that the contract signing protocol proposed in [18] is inherently flawed, regardless of the behavior of the TTP. They refer to this as *resolve impossibility*. We use abort chaining to derive a lower bound on the number of messages in multi-party contract signing protocols.

The contribution of the present paper is threefold. First of all, we model contract-signing protocols as sequences of numbers, which we call *signing sequences*. In this simple model, fairness attacks can be studied independent of cryptographic considerations. The fairness requirement provides boundary conditions on the behavior of the TTP. An abort-chaining attack can be described as a marking of the signing sequence in such a way that it contradicts these boundary conditions, which implies *resolve impossibility* of the modeled protocol. Absence of abort-chaining attacks thus boils down to proving that such a marking does not exist.

Our second contribution is to relate the problem of finding a marking in a signing sequence to the (open) combinatorial problem of finding the shortest sequences containing all permutations of a finite set as subsequences. This problem was stated in 1972 [12] and several authors have provided algorithms to produce such sequences of length $\alpha^2 - 2\alpha + 4$,

where $\alpha > 2$ is the number of elements in the set. This is conjectured to be optimal in the sense that these algorithms produce sequences of minimal length. We prove that this lower bound can be transformed into a lower bound on the length of a signing sequence, and thus on the number of messages in a multi-party contract signing protocol: every protocol consisting of α^2 or fewer messages is unfair, with $\alpha > 2$ being the number of involved signers. For the class of *balanced protocols*, in which all signers roughly send the same number of messages, this bound is provably tight.

The third contribution of this paper is the application of these results to develop novel multi-party contract signing protocols. We show for the case of three signers how the algorithms for generating short permutation sequences can be used to construct a protocol which is shorter than currently known three-party contract signing protocols.

The paper is structured as follows. In Section 2, we discuss literature related to MPCS and our approach. Multi-party contract signing protocols, abort chaining, and our assumptions are explained in Section 3. The interpretation of MPCS protocols as signing sequences is described in Section 4 and the step towards the problem of short permutation sequences is taken in Section 5. This section also contains our main results. In Section 6 we prove the main conjecture for the class of balanced MPCS protocols. An example is presented in Section 7 to illustrate our results. Finally, we summarize the paper and indicate possibilities for future research in Section 8.

2. Related work

The standard reference for the optimistic family of fair exchange protocols is Asokan's thesis [2]. Multi-party optimistic contract signing (or, more generally, fair exchange) protocols have not received as much attention as their two-party variants. Nonetheless, there are several notable exceptions, for instance, the work of [4] for synchronous protocols and [7], [6], [18], [10], [21] for asynchronous protocols. We focus on optimistic asynchronous protocols. One can roughly divide the asynchronous protocols into two groups. One contains the protocol proposed in [7] and improved in [6], and the other contains the protocol from [18], broken and fixed for four parties in [10], and later broken and shown to be beyond repair in [21]. The state of the art in the asynchronous class of protocols has been set by Mukhamedov and Ryan [21], whose protocol is more efficient than [6], and does not have the fatal vulnerability of [18]. This protocol requires $2(\alpha - 1)(\lceil \frac{\alpha}{2} \rceil + 1)$ messages to be exchanged in the main protocol, with α being the number of signing parties involved. This bound pertains to the case where piggybacking is used to reduce the number of messages in the protocol.

Regarding the proof technique, Pfitzmann et al.'s proof that four is the optimal number of messages for two-party contract signing is based on a careful analysis of all the possibilities that may arise if the number of messages exchanged in the main protocol is less than four [23]. Such an exhaustive analysis is infeasible in multi-party settings. A contribution of this paper is to encode multi-party contract signing as an abstract mathematical problem which can be understood and investigated in isolation. Another possible way of proving lower bounds on the number of messages needed to perform a certain distributed task is based on extracting the least amount of knowledge which is necessary to be acquired by the participants, as done for instance in [11].

We remark that we only consider purely asynchronous communication. Therefore, our analysis is orthogonal to those lower bound results which are obtained by assuming the existence of broadcast channels or various degrees of synchrony in the communication system, as in [24], [8].

3. Optimistic multi-party contract signing

In this section we give a conceptual description of the multi-party contract-signing problem. This rather high level of abstraction is sufficient to describe abort-chaining attacks. We mention explicitly that the development of a fully formal semantics of MPCS protocols and the required properties is not a goal of this paper. Such a formalization, while highly desirable, is independent of the contribution of this paper and we consider it a research question on its own. Given the assumptions and restrictions mentioned in this section, the notion of *signing sequences* is for our goals the most appropriate and elegant formalization of the problems under study.

3.1. Communication and intruder model

We assume a fully connected communication network. Communication between signers is asynchronous and messages can get lost or be delayed arbitrary long. In order to simplify our reasoning, we will not assume a general Dolev-Yao style intruder model [13] for the communication between the signers. We consider the problem of delivering secret and authentic messages as orthogonal to the current problem setting. The communication channels between signers and the TTP are assumed to be resilient, which means that the messages sent over these channels are guaranteed to be delivered eventually and without modifications. Various cryptographic techniques can be used to guarantee the integrity of the transmitted messages against a computationally bounded attacker. Guaranteed delivery, however, requires the assumption that communication channels are *fair lossy*, i.e. any message inserted at one end of the channel an infinite number of times, is delivered to the other end of the channel an infinite number of times. Given a fair lossy channel, which may duplicate and reorder messages, then, essentially,

retransmission and tagging allow us to construct a reliable FIFO channel on top of the fair lossy channel, see Stenning's protocol [25] and [20, chapter 22]. Assuming fair lossy channels amounts to assuming a limited attacker who cannot entirely disconnect participant processes.

In view of the abstraction of cryptographic techniques for delivering secret and authentic messages, the following data items will be sufficient for the class of protocols considered in this paper. The properties assumed in the following definition can be realized using cryptographic primitives such as *private contract signing* [17], [21].

Definition 1.

1) *A* contract $\mathrm{contr}(C, TTP, p_1, \ldots, p_\alpha)$ *is a function of a contract text C, a TTP, and signers* $p_1,\ldots,p_\alpha$.
2) *A* level l promise $\mathrm{prom}_l(c,i)$ *is a string uniquely identifying a contract c, signer* p_i*, and level l with the properties that*
   - *Only* p_i *can generate* $\mathrm{prom}_l(c,i)$ *for any l.*
   - *Everybody can verify that* $\mathrm{prom}_l(c,i)$ *identifies the contract c, signer* p_i*, and level l.*
3) *A* signature $\mathrm{sig}(c,i)$ *is a string uniquely identifying a contract c and signer* p_i*. The string has the following properties*
   - p_i *can generate* $\mathrm{sig}(c,i)$.
   - *The TTP can generate* $\mathrm{sig}(c,i)$ *from* $\mathrm{prom}_l(c,i)$ *for any l.*
   - *Nobody else can generate* $\mathrm{sig}(c,i)$.
   - *Everybody can verify that* $\mathrm{sig}(c,i)$ *identifies the contract c and signer* p_i.
4) *A* fully signed *contract is a set* $\{\mathrm{sig}(c,1),\ldots,\mathrm{sig}(c,\alpha)\}$*, where c is a contract.*

3.2. Optimistic contract-signing protocols

The goal of a multi-party contract-signing protocol is to fairly construct a fully signed contract such that, even if one or more signers misbehave, either all *honest* signers obtain a fully signed contract or none of the signers obtain a signed contract [3], [6]. Honest signers are those agents that exhibit no other behavior than prescribed by the protocol. All other agents, except for the TTP, are considered dishonest.

In this paper, we consider *optimistic contract-signing protocols*. Such protocols consist of two sub-protocols, viz. the *main* protocol and the *resolve* protocol. The main protocol governs the regular exchange of promises and signatures between the signers. Upon termination of the main protocol, all signers have obtained a fully signed contract. *We assume that the messages of the main protocol are totally ordered, thereby ignoring protocols with interleaved or parallel messages.* The main protocol does not involve the TTP. The TTP is not even aware of the fact that the protocol is executed, unless her help is requested by one of the signers to resolve a problem. As stated in definition 1, we assume that the TTP (and nobody else) has the capability to transform a collection of promises of all signers into a fully signed contract. She may use this capability in the resolve protocol. The resolve protocol is a two message protocol between a signer and the TTP. The signer sends all promises and signatures in its possession to the TTP. The TTP answers with *abort* or with a fully signed contract. We will discuss the TTP's decision procedure between the two answers in the resolve protocol in the following section.

We summarize our assumptions in Table 1.

Table 1. Assumptions.

- Asynchronous communication.
- Fully connected network.
- Lossy channels between signers.
- Resilient channels between TTP and signers.
- Signer can verify promises without contacting TTP.
- Only TTP can transform set of promises into valid contract.
- Signers execute totally ordered protocol.
- Resolve request contains entire history of requesting signer.
- All messages are integrity-protected.

3.3. Fairness, timeliness, and the TTP's decision

Apart from satisfying the functional requirement of constructing a fully signed contract, contract-signing protocols must satisfy three security requirements, viz. *fairness*, *timeliness*, and *abuse-freeness*.

In the general setting of fair-exchange protocols, fairness means that none of the parties involved in the protocol can gain advantage over any of the other honest parties taking part in the protocol. For contract-signing protocols this is made more precise by requiring that no honest signer should be left in the position of having sent another signer his signature on the contract, without being able to obtain a fully signed contract [21].

Definition 2. *A contract-signing protocol is* fair *for the signers if, for every execution of the protocol, either all honest signers terminate with a fully signed contract in their possession or none of the signers can obtain a signature of an honest signer as a consequence of the protocol execution.*

Timeliness ensures that no signer can force another signer to wait for any length of time [5], or, phrased differently, every signer has some recourse to prevent endless waiting [21].

Definition 3. *A contract-signing protocol satisfies* timeliness *if every execution of every honest signer eventually terminates successfully.*

Successful termination in Definition 3 refers to the fact that the signer must be in an end state rather than a deadlock state.

A signing protocol is said to be abuse-free if at any stage of the protocol, it is impossible for any signer to prove to an outside challenger that he has the power to

choose between completing the contract-signing protocol and aborting it [17]. Although abuse-freeness is an important security requirement for contract-signing protocols, it will not play a role in our observations on message minimality. Henceforth, we will only focus on fairness and timeliness.

Whenever a signer has to wait (too long) for an incoming message, he may stop the main protocol and start the resolve protocol with the TTP by issuing a *resolve request* which includes the signer's full communication history. This means that it contains all promises and signatures sent and received by this signer. After having started the resolve protocol, honest signers are supposed to never continue with the main protocol. Upon termination of the resolve protocol, the signer has either received a *resolve reply* (containing a fully signed contract) from the TTP, or he has received an *abort reply*. In both cases an honest signer terminates the execution of the protocol.

The decision of the TTP to issue an abort reply or a signed contract depends on the information provided by the requesting signer and by all previous resolve requests. The TTP assumes that all signers are honest unless this contradicts the information received in any of the resolve requests. This is her only source of information with respect to honesty.

A signer's ability to start the resolve protocol whenever he has to wait for a message from another signer is a consequence of the timeliness requirement. This requirement also implies that the TTP must reply to resolve requests in finite time. Resolution of requests in finite time implies that in *the worst case* the TTP needs to decide between a resolve reply and an abort reply without the possibility to wait for further resolve requests or to contact other signers. If the TTP has to wait for an answer of signer p in order to reply to a request of signer q, progress of q would depend on the willingness of p to answer in a timely fashion. In particular, a protocol in which the TTP uses broadcast cannot satisfy timeliness. Thus we may assume without loss of generality that the TTP immediately decides and replies to resolve requests. This leads to the following decision rules which any decision procedure under the above assumptions must follow: TTP0 and TTP1 follow from the timeliness requirement, TTP2 and TTP3 from the fairness requirement.

- TTP0 All signers are considered honest, until the TTP's information shows that a signer deviated from the protocol specification, for instance by participating in the main protocol after having issued a resolve request.
- TTP1 If the TTP does not have enough information to issue a fully signed contract, she answers with an abort message.
- TTP2 If a signature has been sent by any currently considered honest signer, then a fully signed contract is sent.
- TTP3 If a previous reply was an abort reply to a signer which is currently considered honest, then an abort reply is sent.

We observe that, in some instances, the decision rules allow the TTP to issue a fully signed contract in spite of earlier abort replies, viz. when previous abort replies were issued to dishonest signers. However, if the TTP ever sends a fully signed contract to a signer she must send a fully signed contract to every subsequent resolve request of a (supposedly) honest signer.

3.4. Abort chaining

From the fairness requirement it follows that if the TTP ever sends an abort reply to a (presumably) honest signer p, she has to send an abort reply to every subsequent resolve request as well, unless, based on new information, p can be proved to be dishonest. In that case she may later decide to overturn the decision and to issue a resolve reply. A complicating factor is that the TTP in the meantime may have sent an abort reply to a signer that is (still) considered honest, implying that she has to stick to the abort decision. This is the problem of *abort chaining* as introduced by Mukhamedov and Ryan [21] to prove their resolve-impossibility result.

Figure 1 contains an example of an MPCS protocol instantiated for five signers. Messages that contain a promise to send a signature are represented by unlabeled arrows. The arrows labeled *sig* denote messages containing one or more signatures. The protocol consists of three rounds, each of which contains a message sequence from signer 1 to signer 5 and back.

The dots in the diagram indicate the points where a signer can issue a resolve request. These are the points where the signer expects an incoming message. Such a dot indicates that the signer has sent the outgoing message at this point in the protocol, but complains to the TTP that he has not received the next expected message. For example, the first ?-dot on the left subfigure indicates that signer 5 has sent the 5th protocol message to signer 4, but has not received the 12th protocol message from signer 4.

These points are labeled with the possible decisions of the TTP when receiving this resolve request. Label A denotes that the TTP must decide to send an abort reply because the resolve request cannot contain sufficient information to construct a fully signed contract. Label R denotes that the TTP must decide to reply with a fully signed contract, because at that point the requesting signer has already sent a contract with his signature to another signer. In the other cases, labeled with ?, the decision of the TTP is not a priori determined. She may have to send an abort reply or a resolve reply based on her current knowledge and history.

The right diagram in Figure 1 shows an abort-chaining attack on the protocol, implying that the protocol is not fair. The attack goes as follows. Signer 4 is dishonest and sends

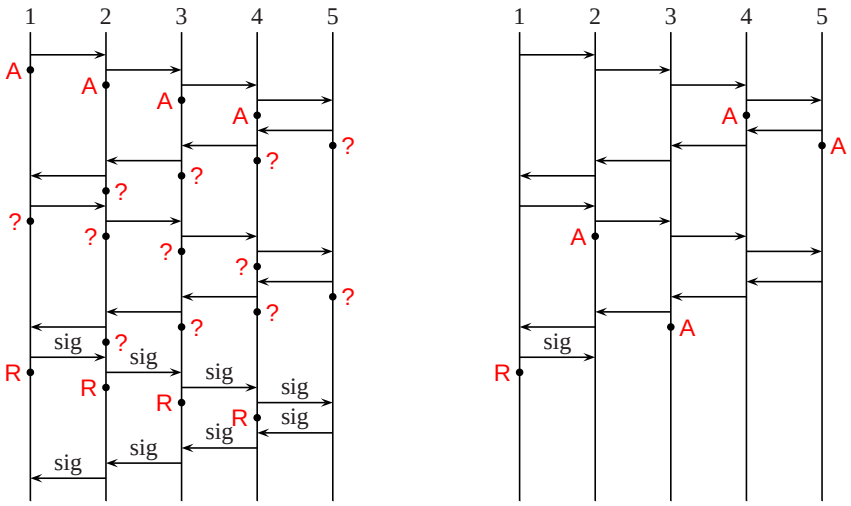

Figure 1. An abstract MPCS protocol indicating enforced abort and resolve decisions (left) and an abort-chaining attack on this protocol (right).

a resolve request to the TTP while still continuing his role in the protocol. Since the TTP assumes that signer 4 is honest until the opposite is proved she has to decide to abort the protocol and sends an abort reply to signer 4. Next, signer 5 decides to send a resolve request. This request is based on the first five messages of the protocol, so the TTP cannot conclude from this request that the previous resolve request by signer 4 was bogus, so she still believes that signer 4 is honest and, consequently, she has to send an abort reply to signer 5, too. However, signer 5 is dishonest as well and continues his execution of the protocol. The next signer to complain is signer 2. Since his resolve request concerns the first 10 messages of the protocol, it is now clear to the TTP that signer 4 has been cheating. However, she cannot overturn her decision to abort the protocol, since in the meantime she has sent an abort reply to signer 5, who is still assumed to be honest. The same situation occurs after signer 3 decides to issue a resolve request. Since the TTP does not know yet that signer 2 is dishonest, she has to be consistent in her choice to send an abort reply. The problem now is that by the time signer 1 complains, he has already sent a signed contract to signer 2 without receiving a signed contract from the other signers. Whichever answer the TTP sends to signer 1, he or signer 3 will be treated unfairly while neither can be shown to be dishonest. Summarizing, this abort chain implies that there cannot be a resolve protocol correctly handling the resolve requests, so the main protocol does not satisfy fairness. For more on the abort chaining attack and an example of its realization on a concrete protocol, see [21].

4. Signing sequences

In this section we define *signing sequences*, which serve to formulate a precise and idealized description of the abort-chaining problem, based on the high-level description provided in the previous section.

Our starting point is the observation that a totally-ordered contract-signing protocol can be represented by a sequence of numbers $\sigma_1, \ldots, \sigma_n$, where σ_i identifies the sending of a message by signer p_{σ_i}.

For instance, the protocol in Figure 1 (left) can be represented by the sequence

$$12345432\ 12345432\ 12345432.$$

The TTP's decision rules allow us to naturally distinguish three phases in the main protocol. We recall that in the resolve protocol the requesting signer sends all promises and signatures in his possession to the TTP. The TTP uses all information received in her decision procedure following the rules TTP0 through TTP3.

Definition 4. *The* initial phase *is the largest sequential union of all messages of the main protocol, starting from the first message, based on which a TTP is forced to send an abort answer to a resolve request. The* end phase *is the set of messages starting with the earliest message in the main protocol in which a signature* $\mathrm{sig}(c, i)$ *is sent and ending with the last message of the protocol. The* middle phase *of the protocol is the set of messages not belonging to the initial phase nor to the end phase.*

We will not consider degenerate protocols in which the initial phase and the end phase have a non-empty intersection. Thus the initial phase contains all messages up to the first promise of the signer who is the last to become active in the protocol (i.e. signer 5 in Figure 1). During this phase, the TTP is forced to decide to abort the protocol upon a resolve request because she does not have enough information to generate a fully signed contract yet. In the middle phase promises are exchanged and the level of commitment is

increased. The end phase starts when the first signed contract is sent by a signer. During this phase, the TTP is forced to decide to resolve the protocol upon a resolve request.

Two observations concerning the shape of the initial and end phase will allow us to define an *idealized* contract signing protocol. Both observations result from our focus on abort-chaining attacks.

Observation 1. In the initial phase, only the *last* appearance of each signer is relevant. For an honest signer appearing several times in the initial phase, a resolve request at any stage in the initial phase will lead to an abort reply from the TTP. For a dishonest signer sending a resolve request before his last appearance in the initial phase only makes it easier for the TTP to disqualify the signer as being dishonest. Consequently, if there are α signers, we can assume that the idealized protocol starts with a sequence of α messages, one for each signer. Then the initial phase of the idealized protocol consists of the first $\alpha - 1$ messages, which are sent by $\alpha - 1$ different signers and the middle phase starts with a message sent by the one remaining signer.

Observation 2. In the end phase, only the *first* appearance of each party is relevant. For any resolve request in this phase, the TTP has to resolve by sending a fully signed contract. The later the resolve request occurs, the more signers can be shown to have participated, making it easier for the TTP to identify cheaters. Therefore, we can assume that in the end phase of the idealized protocol every signer sends exactly one message, so the end phase of the idealized protocol has length α.

In the remainder of this paper, we will analyze the minimal number of messages of an idealized protocol. The two observations above enable us to select only those events from the protocol which are essential for the abort-chaining analysis. In order to translate this result back to full MPCS protocols, the events that were deleted during idealization have to be taken into account. The following observation shows that this means that we then have to add $(2\alpha - 2) - \alpha = \alpha - 2$ messages.

Observation 3. The end phase of an MPCS protocol must consist of at least $2\alpha - 2$ messages. This can be seen as follows. In the end phase all signers have to send their signature. This costs at least α messages. After the last signer has sent his signature, there are at least $\alpha - 2$ messages needed to distribute his signature to all other signers.

Following Observations 1 and 2 we can represent a totally-ordered MPCS protocol, such as the one in Figure 1, more abstractly as a sequence of numbers

$$1234 \mid 543212345432 \mid 12345$$

where number n corresponds to the sending of a message by signer p_n. The separators indicate the three phases of the protocol. The end sequence, 12345, is in accordance with Observation 2, stating that the idealized end phase has length $\alpha = 5$. We enhance the notation by including the resolve requests as dot marks above the numbers. For instance, in the attack in Figure 1, the dot marks appear as follows:

$$1\dot{2}3\dot{4} \mid \dot{5}4\dot{3}21\dot{2}345\dot{4}\dot{3}2 \mid \dot{1}2345.$$

More precisely, a dot above number n means that after having sent the current message, signer p_n decides to issue a resolve request. Using this notation we can now give a simple characterization of an abort-chaining attack. A dot marking can be interpreted as a successful attack if and only if it satisfies the following properties.

- P1 A number with a mark cannot be marked anywhere else in the sequence.
  This follows from the fact that if a signer sends a second resolve request, the signer will be considered dishonest and his request will be ignored.
- P2 The left-most mark must be in the initial phase of the protocol (i.e. left of the first separator).
  This follows from the fact that the first resolve request needs to be answered with an abort reply for an abort-chaining attack to be possible.
- P3 The right-most mark must be in the end phase of the protocol (i.e. right of the second separator).
  The reason is that the TTP cannot abort a resolve request of an honest signer in this phase of the protocol, since the signer already signed the contract.
- P4 Between two successively marked numbers, say n_1 and n_2, there must be no occurrence of n_1.
  If there would be an occurrence of n_1 between two consecutive resolve requests of n_1 and n_2, then the resolve request of n_2 would reveal that n_1 behaved dishonestly. This is because the TTP can infer from resolve request n_2 that n_1 has continued with the normal course of the protocol after having sent a resolve request. In such a case the TTP could immediately overturn an earlier abort decision for n_1.

Henceforth, we will assume that A (the set of signers) is a finite set of size α. Let A^* denote the set of all finite sequences $\sigma = (\sigma_1, \ldots, \sigma_n)$ over A. A sequence σ' is a *subsequence* of σ if σ' can be obtained by erasing zero or more symbols from σ.

The observations above are formalized as follows.

Definition 5. *Let $\sigma = (\sigma_1, \ldots, \sigma_n) \in A^*$. Then σ is a signing sequence if $n \geq 2\alpha$ and if it starts and ends with a permutation of A. The prefix of length $\alpha - 1$ of σ is called the initial sequence of σ and the suffix of length α is called the end sequence of σ. The sequence remaining after removing the initial and end sequence from σ is called the middle sequence of σ.*

The four conditions for successful abort-chaining attacks can now be stated as follows.

Definition 6. *Let $\sigma \in A^*$ be a signing sequence of length n. A subsequence $\sigma_{f(1)}, \ldots \sigma_{f(k)}$ of σ is called an abort-chaining subsequence (AC subsequence for short) if the following holds:*

1) $\forall_{p \neq q}\ \sigma_{f(p)} \neq \sigma_{f(q)}$;
2) $f(1) < \alpha$;
3) $f(k) > n - \alpha$;
4) $\forall_p\ \sigma_{f(p)} \notin \bigcup_{f(p)<j<f(p+1)} \sigma_j$.

A signing sequence σ which has an AC subsequence will be called *unfair*. A signing sequence which is not unfair will be called *fair*. The relation between protocols satisfying the fairness requirement and fair signing sequences is stated in the following Lemma.

Lemma 1. *Every MPCS protocol which gives rise to an unfair signing sequence does not satisfy the fairness requirement.*

Proof: The lemma follows from the formalization leading to Definition 6 as follows. An unfair signing sequence satisfies the conditions in Definition 6. These conditions are equivalent to the properties P1 through P4. By property P2, the TTP has issued an abort reply to a signer. By repeated application of properties P1 and P4, the TTP has issued an abort or fully signed contract to a signer while the previous resolution-seeking signer is deemed honest. Since the first answer was an abort reply, all the following answers are an abort reply. By property P3 and rule TTP2, the TTP has issued a fully signed contract to a signer. Since the previous resolution-seeking signer is honest and the current signer is honest, fairness cannot be satisfied. □

Note that a fair signing sequence σ does not necessarily give rise to a fair protocol, because there may be attacks on the fairness property of a protocol which are not abort-chaining attacks, while unfair signing sequences are limited to the existence of abort-chaining attacks.

It follows that the number of messages in fair contract-signing protocols investigated in this paper is necessary to achieve fairness, but it may in general not be sufficient.

5. Relation to an open problem

An open combinatorial problem is to find the shortest sequence over a set A which contains every permutation of elements of A as a subsequence. For instance, if $A = \{1, 2, 3\}$, the shortest such sequence is 1213121. This problem has been listed as a research problem in [12, Problem 36] where it is attributed to R.M. Karp. Solutions by Newey [22], Adleman [1], and Galbiati and Preparata [16] show that the length $l(\alpha)$ of such a sequence is at most $\alpha^2 - 2\alpha + 4$ for $\alpha = |A| \geq 3$.

In this section we show that the problem of finding the shortest fair signing sequence is equivalent to the problem of finding the shortest sequence containing all permutations of elements in A as a subsequence. For the remainder of this and the following section we will assume that $|A| \geq 3$.

Theorem 1. *A signing sequence σ is fair if and only if the sequence consisting of its middle and end sequence contains all permutations of A as a subsequence.*

Proof: Let σ' be the sequence consisting of the middle and end sequence of σ. Assume σ is fair. Suppose towards a contradiction that p is a permutation of A which is *not* a subsequence of σ'. Then by Lemma 2 below, p can be transformed into an AC subsequence of σ contradicting fairness of σ.

Conversely, assume that σ' contains all permutations of A as a subsequence and suppose towards a contradiction that $s = (\sigma_{f(1)}, \ldots, \sigma_{f(k)})$ is an AC subsequence of σ. We may assume without loss of generality that $f(2) \geq \alpha$ and $f(k-1) \leq n - \alpha$. Since σ' contains *all* permutations of A as a subsequence, it must contain s as a subsequence $(\sigma_{g(1)}, \ldots, \sigma_{g(k)})$. By Condition 4 of definition 6, it follows that $g(i) \leq f(i)$ or $g(i) > f(i+1)$ for $i = 1, \ldots, k-1$. However, since $(\sigma_{g(1)}, \ldots, \sigma_{g(k)})$ is a subsequence of σ', it follows that $g(1) \geq \alpha > f(1)$, thus $g(1) > f(2)$. Since $g(i+1) > g(i)$, it follows inductively that $g(i) > f(i+1)$ for $i = 1, \ldots, k-1$. This implies that $g(k) > g(k-1) > f(k)$ which is a contradiction, since $g(k) > f(k) > n - \alpha$, $\sigma_{g(k)} = \sigma_{f(k)}$, and the end sequence is a permutation of A. □

The lemma below is used in the proof of theorem 1.

Lemma 2. *A signing sequence σ has an AC subsequence if there is a permutation $p = (p_1, \ldots, p_\alpha)$ of A such that p is not a subsequence of the sequence consisting of the middle and end sequence of σ.*

Proof: We construct an AC subsequence $(\sigma_{f(j-1)}, \ldots, \sigma_{f(\alpha)})$ of σ by computing its indices $f(j-1), \ldots, f(\alpha)$ backwards, starting from $f(\alpha)$.

Since σ is a signing sequence, there is a unique element $\sigma_i = p_\alpha$ in the end sequence. Let $f(\alpha)$ be the index of that element, that is,

$$f(\alpha) = \max\{i \mid \sigma_i = p_\alpha\}.$$

Consider the longest suffix $(p_j, p_{j+1}, \ldots, p_\alpha)$ of p which is a subsequence of the middle and end sequence of σ. (Since p itself is not a subsequence, it follows that $j > 1$.)

Let $f(j), \ldots, f(\alpha)$ be an increasing sequence such that for $j \leq i < \alpha$

$$f(i) = \max\{\iota \mid \iota < f(i+1), \sigma_\iota = p_i\}. \tag{1}$$

Such a sequence exists, because $(p_j, p_{j+1}, \ldots, p_\alpha)$ is a subsequence of the middle and end sequence of σ.

Since $(p_j, p_{j+1}, \ldots, p_\alpha)$ is the longest possible subsequence and σ is a signing sequence, it follows that there exists $f(j-1) < \alpha$ with $\sigma_{f(j-1)} = p_{j-1}$. (Recall that the first α elements of σ are a permutation of A.)

We show that $(\sigma_{f(j-1)}, \ldots, \sigma_{f(\alpha)})$ is an AC sequence of σ by verifying all conditions of the definition:

- Condition 1 is satisfied since p is a permutation.
- Condition 2 is satisfied since $f(j-1) < \alpha$.
- Condition 3 is satisfied since $f(\alpha)$ was chosen such that $f(\alpha) > n - \alpha$.
- Condition 4 is satisfied by equation 1. □

To illustrate the preceding Lemma and Theorem, consider the signing sequence σ given by

$$1234 \,\big|\, 543212345432 \,\big|\, 12345.$$

Its middle and end sequence is the sequence

$$543212345432 12345.$$

The permutation 45231 does not occur as a subsequence in the middle and end sequence, thus we can find an AC subsequence by marking the first occurrence of the elements of 45231 backwards in the signing sequence:

$$1234 \,\big|\, 543212345432 \,\big|\, \dot{1}2345$$
$$1234 \,\big|\, 54321234\dot{5}432 \,\big|\, \dot{1}2345$$
$$1234 \,\big|\, 5432\dot{1}2345\dot{4}32 \,\big|\, \dot{1}2345$$
$$1234 \,\big|\, \dot{5}432\dot{1}2345\dot{4}32 \,\big|\, \dot{1}2345$$
$$\dot{1}234 \,\big|\, \dot{5}432\dot{1}2345\dot{4}32 \,\big|\, \dot{1}2345$$

Conversely, the permutation 15234 appears as a subsequence of the middle and end sequence emphasized in boldface:

$$5432\mathbf{1}2345\mathbf{4}32\mathbf{1}2\mathbf{34}5.$$

By the construction in the proof of Theorem 1, the permutation sequence 15234 cannot be an AC subsequence of σ, because elements of any valid marking will always trail behind the permutation sequence:

$$\dot{1}234 \,\big|\, \dot{5}432\dot{1}2345\dot{4}32 \,\big|\, 12\mathbf{34}5$$

By Condition 4 of Definition 6, the 4 in the end sequence cannot be marked because of the boldface 3 in the end sequence.

Corollary 1. *Let n be the length of a shortest fair signing sequence over A. Then there is a sequence of length $n-\alpha+1$ containing all permutations of A as subsequences.*

Proof: Let σ be the shortest fair signing sequence over A. The initial sequence of σ has length $\alpha-1$, thus the length of the middle and end sequence is $n-(\alpha-1)$. By Theorem 1, this is a sequence containing all permutations of A. □

Corollary 2. *Let λ be the length of the shortest sequence containing all permutations of A as subsequences. Then the shortest fair signing sequence has length at least $\lambda+\alpha-1$ and at most $\lambda+2\alpha-3$.*

Proof: Let σ' be the shortest sequence of length λ containing all permutations of A as subsequences. Let $p_1, \ldots, p_{\alpha-1} \in A \setminus \{\sigma'_1\}$ be pairwise distinct.

- Suppose that the first or the last α elements of σ' are a permutation of elements in A. (That is, $\{\sigma'_{\lambda-\alpha+1}, \ldots, \sigma'_\lambda\} = A$ or $\{\sigma'_1, \ldots, \sigma'_\alpha\} = A$.)
  Since the reverse sequence of σ' contains all permutations of A as a subsequence if and only if σ' does, we may, in this case, assume without loss of generality, that σ' ends with a permutation of elements in A. By Theorem 1, $\sigma = (p_1, \ldots, p_{\alpha-1}, \sigma'_1, \ldots, \sigma'_\lambda)$ is a fair signing sequence, of length $\lambda+\alpha-1$. A fair signing sequence shorter than σ would, by Corollary 1, contradict the minimal length of σ'.
- Suppose that neither the first nor the last α elements of σ' are a permutation of elements in A.
  Then, for $p'_1, \ldots, p'_{\alpha-2}$ such that $\{p'_1, \ldots, p'_{\alpha-2}, \sigma'_\lambda, \sigma'_{\lambda-1}\} = A$, the sequence $\sigma = (p_1, \ldots, p_{\alpha-1}, \sigma'_1, \ldots, \sigma'_\lambda, p'_1, \ldots, p'_{\alpha-2})$ is a fair signing sequence by Theorem 1. Its length is $\lambda+2\alpha-3$. □

Remark 1. We could strengthen the last two corollaries if we knew that the shortest sequence containing all permutations of A as subsequences starts or ends with a permutation of A. In that case, the difference between the lengths of shortest fair signing sequences and shortest sequences containing all permutations of A as subsequences would always be $\alpha-1$.

Newey [22] and Adleman [1] have constructively shown that there are sequences of length $\alpha^2-2\alpha+4$ containing all permutations of A as subsequences. These constructions are such that the sequences start with a permutation of A. We can therefore set $\lambda = \alpha^2-2\alpha+4$ in Corollary 2 and obtain our main theorem.

Main Theorem. *There is a fair signing sequence in A^* of length $\alpha^2-\alpha+3$.*

Proof: Apply the proof of Corollary 2 to Newey's or Adleman's sequences. □

Remark 2. A fair signing sequence of length $\alpha^2-\alpha+3$ can be transformed back into a MPCS protocol of length α^2+1. However, this transformation does not automatically imply fairness of the resulting MPCS protocol, but merely resistance to abort chaining attacks.

Newey's and Adleman's construction and the conjecture [22], [19] that $\alpha^2-2\alpha+4$ is the shortest possible length for any sequence containing all permutations of A as subsequences imply the following conjecture.

Conjecture 1. *For $\alpha \geq 3$, all signing sequences $\sigma \in A^*$ with $|\sigma| < \alpha^2-\alpha+3$ are unfair.*

Main Conjecture. *For $\alpha \geq 3$, all MPCS protocols with fewer than $\alpha^2 + 1$ messages are unfair.*

The Main Conjecture follows from Conjecture 1 by adding the minimal length of the end phase as derived in Observation 3 minus α, which is the length of the end phase in the abstract signing sequences.

In the next section, we will prove the Main Conjecture for the class of MPCS protocols in which all parties send equally many messages in the middle phase of the protocol. We refer to such protocols as being *balanced*. [1]

6. Balanced protocols

According to the Main Conjecture, stated in the preceding section, an MPCS protocol in which at most α^2 messages are sent cannot be fair. We prove that the conjecture is true for protocols in which no signer sends more than $\alpha-2$ messages during the middle phase of the protocol.

Theorem 2. *If $\alpha \geq 3$ and σ is a signing sequence in whose middle sequence no element appears more than $\alpha-2$ times, then σ is unfair.*

Proof: If a sequence contains all permutations of elements of a set A as subsequences, then there must be an element of A appearing at least α times. This has been observed by Newey [22, Observation 2.12] and can be shown by mathematical induction on the number of elements of A. It follows from this observation that a sequence in which each element of A appears less than α times cannot contain all permutations of A. The Theorem now follows from the equivalence in Theorem 1. □

We now have the following immediate corollaries.

Corollary 3. *MPCS protocols with at least three parties in whose middle phase no party sends more than $\alpha-2$ messages are unfair.*

Corollary 3 can be used to prove lower bounds for a variety of MPCS protocols that satisfy some regularity conditions. For instance, if every party sends equally many and at most $\alpha - 2$ messages during the middle phase of an MPCS protocol, the protocol is unfair. In this case we have $\alpha^2 - 2\alpha$ messages in the middle sequence. Under the assumption that a minimal number of messages will be exchanged in the initial and end phase of the protocol, the initial phase contains $\alpha - 1$ messages and the end phase contains $2\alpha-2$ messages. Thus we have a total of $\alpha^2+\alpha-3$ messages and we obtain the following corollary.

Corollary 4. *MPCS protocols with at least three parties and of at most $\alpha^2+\alpha-3$ messages in whose middle phase every party sends equally many messages are unfair.*

1. The notion of *balanced* as used here pertains to even distribution of messages exchanged in an MPCS protocol; this should not be confused with *balanced MPCS protocols* as introduced in [9].

7. A minimal protocol

In this section, we build upon the previous sections to design a three-party optimistic contract signing protocol with minimal number of messages. The presented protocol exchanges only 10 messages in the main protocol, hence being more efficient than any existing protocol for this purpose (the protocol of Mukhamedov and Ryan [21] exchanges 12-messages in the main protocol).

The main purpose of this section is to illustrate the concepts defined before and to show their possible application. Thus, a full and formal verification of this protocol is outside the scope of this paper. The level of formality and precision achieved in the current security analysis is comparable with the level of formality achieved in the proofs of state-of-the-art and well-established protocols, such as in [21]. We leave a model-checker based verification of the current protocol for future research.

To construct the protocol the following steps are taken:

- A sequence which contains all permutations of numbers in $\{1,2,3\}$ is constructed, e.g. using Adleman's algorithm [1]. This serves as the middle and (idealized) end phase of the signing sequence. One such sequence is $3123 \mid 123$, of length $\alpha^2 - 2\alpha + 4$ which evaluates to seven when $\alpha = 3$. It has been shown in [22] that seven is indeed the length of the shortest sequence containing all permutations of $\{1,2,3\}$.
- The initial phase and the end phase are completed according to Observations 1 and 2. We thus get $12 \mid 3123 \mid 123$ as the complete signing sequence and the exchange pattern shown in Figure 2 for the complete protocol.

We specify the contents of messages, TTP's logic, etc., similar to the protocol of [21]. These are briefly explained below; all other terms are according to Definition 1.

We write $[M]_i$ for message M signed by entity i, where $i \in \{1,2,3,T\}$ with T standing for the TTP. [2] We assume that M can be extracted from $[M]_i$, and that all agents check the validity of the received signatures. A message which carries a bogus signature is destroyed by the recipient and is considered as not being received. Messages exchanged in the protocol are assumed to uniquely identify the involved signers, their positions in the network, the identity of the trusted party, and the contract text. We write $c = \mathrm{contr}(C, TTP, p_1, \ldots, p_\alpha)$ for the complete description of this information. The contents of messages are specified in Figure 2.

The intuition is that signers start by giving a level 1 promise to sign the contract, then they increase the level of their promise to level 2, and finally they release their

2. Note that these signatures are used to authenticate messages, and should not be confused with a party's signature on a contract.

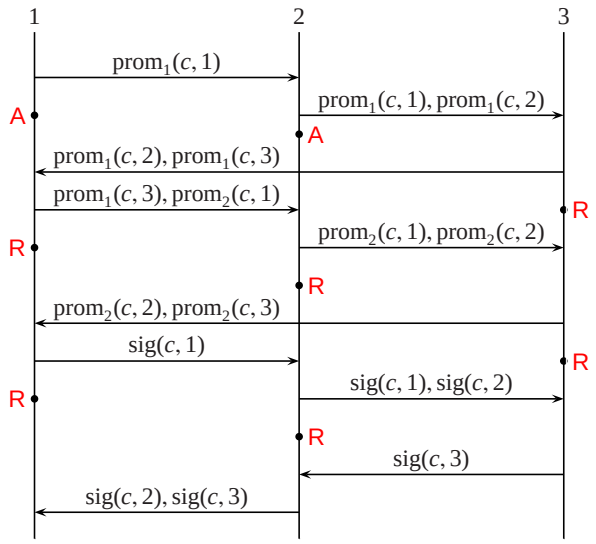

Figure 2. Three-party contract signing with 10 messages in the main protocol

signature on the contract (that is $\text{sig}(c, i)$, for $i \in \{1, 2, 3\}$). A *signed contract* is the set

$$\{\text{sig}(c, 1), \text{sig}(c, 2), \text{sig}(c, 3)\}$$

The TTP is assumed to be able to transform $\text{prom}_l(c, i)$ to $\text{sig}(c, i)$, for any level l and any $i \in \{1, 2, 3\}$; no one else except i has this ability. Cryptographic techniques to realize such transformations are prevalent in the literature, e.g. [21].

If an expected message does not arrive, or the received message does not conform to the protocol, signer i may simply quit the protocol or resort to the TTP, depending on where i is in the main protocol. A signer quits the exchange if and only if he has not received any message in the main protocol. That is, signers 2 and 3 can simply quit the exchange if they do not receive $\text{prom}_1(c, 1)$ and $\text{prom}_1(c, 1), \text{prom}_1(c, 2)$ in time, respectively. In all other cases, signers have to resort to the TTP by sending the following message:

$$i \rightarrow T : [\text{dispute}, i, H_i, c]_i$$

Here, dispute is a reserved keyword indicating that the signer is disputing an exchange, and H_i is the *history* of signer i. The history of i at each stage of the protocol is the set of messages that he has sent or received so far in the current execution. For instance, $H_1 = \{\text{prom}_1(c, 1), \text{prom}_1(c, 2), \text{prom}_1(c, 3), \text{prom}_2(c, 1)\}$ for signer 1, after sending the 4^{th} message and before receiving the 6^{th} one. By sending his history, a signer attests his position in the main protocol to the TTP and gives hints to the TTP about the positions of other signers. Nevertheless, a malicious signer can obviously lie about his history, e.g. by pretending to be behind the actual state of the protocol.

Note that when contacting the TTP, signers do not explicitly specify whether they ask for an abort A or a resolve R outcome. It is up to the TTP to decide which of these must be the outcome of the dispute. The abort and resolve symbols shown in Figure 2 merely indicate the decision of the TTP in case the first dispute is raised at the corresponding point.

For each contract signing session, uniquely identified with c, the TTP maintains a tuple $\langle c, status \rangle$ in her persistent, secure database *DB*. Upon receiving a dispute request $[\text{dispute}, i, H_i, c]_i$, the TTP checks whether her identity is mentioned in c as the trusted entity in the exchange, whether all the signatures contained in the history are genuine, and whether the history is valid given the number of signers and their position in the network. If all these tests pass, the TTP checks whether $\langle c, * \rangle \in DB$:

- $\langle c, * \rangle \notin DB$: A new exchange is being disputed. If H_i corresponds to a point with an A symbol in Figure 2, then the TTP sends back

  $$T \rightarrow i : [\text{A}, c]_T$$

  and stores $\langle c, (i : H_i) \rangle$ in *DB*. The pair $(i : H_i)$ helps the TTP to remember at what position i claims to be at the moment of raising the dispute (below, we see how this information is used). On the other hand, if H_i corresponds to a point with an R in Figure 2, then the TTP computes $S = \{\text{sig}(c, 1), \text{sig}(c, 2), \text{sig}(c, 3)\}$ and sends back

  $$T \rightarrow i : S$$

  and stores $\langle c, S \rangle$ in *DB*. Note that all H_i that correspond to an R in Figure 2 contain $\text{prom}_1(c, 1), \text{prom}_1(c, 2), \text{prom}_1(c, 3)$. Therefore, the TTP is able to compute S from such histories.
- $\langle c, status \rangle \in DB$: In case $\langle c, S \rangle \in DB$, for a set of signatures S, then the TTP simply replies as:

  $$T \rightarrow i : S$$

  Note that this corresponds to the case the exchange has been disputed before, and has been settled to a resolve. We split the case the exchange has been considered aborted before, i.e. $\langle c, (j : H_j) \rangle \in DB$, into two different situations: [3]
  - The current H_i corresponds to an abort point in Figure 2. Then, the TTP sends back

    $$T \rightarrow i : [\text{A}, c]_T$$

    and appends $(i : H_i)$ to the *status* associated to c, i.e. the TTP stores $\langle c, (j : H_j); (i : H_i) \rangle$ in *DB*.
  - The current H_i corresponds to a resolve point in Figure 2. This case is more complicated, as the TTP may realize that some of the past disputes were malicious. The key idea here is that the TTP may overturn her abort decision, if she can deduce

3. As we will see in the following, the *status* associated to c in *DB* may contain a list of histories of the form $\langle c, (j : H_j); \cdots; (i : H_i) \rangle$.

from the current history H_i that all the previous disputes (resulted in abort) were malicious. A previous abort by signer j is deemed malicious iff the current history H_i contains a message signed by j which shows that j has continued the protocol after resorting to the TTP (the position of j at the time he resorted to the TTP is stored in *status* of c as j's history). If the TTP finds out that *all* the previous disputes were malicious, then she sends back

$$T \rightarrow i : S$$

and replaces the entry in *DB* corresponding to c with $\langle c, S \rangle$. Otherwise, the TTP sends back

$$T \rightarrow i : [\mathsf{A}, c]_T$$

and appends $(i : H_i)$ to the *status* associated to c in *DB*.

Remark 3. The protocol presented in this section is more efficient than existing protocols for the same purpose. It has however a feature, which might limit its usability in certain scenarios. The exchange pattern of the protocol (and thereby the algorithms executed by the principals) depend on the number of involved parties. For example, if another party is added to the exchange, to ensure efficiency, we need to devise a new exchange pattern, e.g., based on the algorithm of [1]. This can be seen as a trade-off between efficiency and scalability of these protocols.

Security analysis. In the following, we give a proof sketch for the security of the protocol described above. Recall that the desired properties of the protocol consist of timeliness, abuse-freeness, and fairness.

• Timeliness is achieved simply because a signer can give up waiting for a message at any time and resort to the TTP (if the signer has not received any message, then he can simply quit the exchange); the TTP will answer to any dispute without waiting for input from other signers. It is worth mentioning that all disputes raised by honest signers will be processed by the TTP. This is due to the assumption (cf. Table 1) that any signer can check whether the signatures he receives are valid without contacting the TTP.

• Abuse-freeness is implied by using *private contract signatures* à la [21] for the signatures (i.e. $[M]_i$) exchanged in the protocol.

• Now, we turn to fairness. Suppose an execution of the protocol has ended, i.e., none of the honest signers are waiting for input. This state is always reachable, due to timeliness of the protocol. Further, assume signer i has received honest signer j's signature on the contract. Then, we show that any honest signer k has received a signed contract, consisting of the signatures of all the involved signers on the contract. We distinguish two possibilities:

1) Signer i has obtained j's signature by receiving a signed contract from the TTP via a resolve protocol. If signer k has not received a signed contract in the main protocol, he must have resorted to the TTP. If k's dispute has been processed after i's request, then according to the TTP's logic, k has also received the signed contract from the TTP, via a resolve protocol. If k's dispute has been processed before i's request, we need to show that the TTP would never send a signed contract to any signer, after sending an abort token to k (or, any honest signer in general). This, however, is clear, as the only case in which the TTP considers a signer as malicious and overturns her abort decision, is the one where the signer proceeds in the main protocol after having resorted to the TTP. Honest signers in general, and k in particular, do not participate in the main protocol after disputing the exchange.
2) Signer i has obtained j's signature (possibly indirectly) from j himself; that is, j has released his signature in the main protocol. Honest signer k either receives a signed contract in the main protocol, or resorts to the TTP. Therefore, k may not receive a signed contract only if the TTP replies with an abort token to k's dispute. The TTP replies with *abort* to an honest signer (in particular k) only if either this honest signer k has disputed the exchange in the initial phase of the protocol, or the TTP cannot grant a resolve due to her earlier abort decisions. The former is impossible, since j would not have released his signature if an honest signer had stopped the main protocol in the initial phase. The latter is a typical instance of abort chaining, to which, according to our theorems, this protocol is not susceptible.

While a full proof of timeliness, abuse-freeness, and fairness is not the aim of the preceding analysis, given a suitable framework, such a proof can be achieved following the sketch outlined above.

8. Conclusions

In this paper, we derived a lower bound on the number of messages in an optimistic asynchronous multi-party contract signing protocol. This lower bound is not of a purely theoretical nature, since our approach has several implications with respect to security.

First of all, our work affects the design of correct and efficient multi-party contract signing protocols. Due to the constructive nature of our approach, we can generate protocols which are optimal with respect to the derived bound. We illustrated this for the case of three signers by constructing a protocol which is shorter than current state-of-the-art solutions. We leave the generalization of this construction, including a full and formal correctness proof as a topic of future research. We conjecture that the protocols obtained in

this way will be correct since the three security requirements follow by construction: abuse-freeness is implied by the use of private contract signing, timeliness follows from the ability to contact the TTP, and fairness follows mainly from the absence of abort-chaining attacks. Indeed, we conjecture that under the given assumptions, with the given structure of the messages, the only source of unfairness can be abort chaining.

The second application to security consists of the possibility to efficiently verify protocols in this domain with respect to abort-chaining attacks. Obvious algorithms to test if a sequence contains all permutations, can be easily found. An interesting question is whether it is possible to improve upon these obvious algorithms.

As stated before, developing a full formalization of MPCS protocols was not the goal of this paper because that would be orthogonal to the results of our research. We consider signing sequences as an elegant and formal model of the problem of abort chaining. Nevertheless, the development of a full formal model of MPCS protocols and their security requirements is an important next step.

There are several other directions in which future work could be carried out. Given a set of regularity requirements imposed by practical considerations, the question of the most efficient MPCS protocol satisfying these requirements arises. Such questions could be treated similarly to the balanced protocols in Section 6. Along the same lines and in view of the present results, the Mukhamedov and Ryan protocol [21] is close to being optimal for an even number of parties, but not for an odd number of parties.

Concerning the fairness property itself, it can be expected that similar methods for reasoning about fairness of multi-party contract signing protocols exist when some of the assumptions shown in Table 1 are relaxed. In particular, it would be interesting to investigate such methods when allowing for concurrency in the protocol execution instead of requiring total order of messages.

Aside from tackling the obvious open problem of proving the main conjecture, showing that for any shortest sequence containing all permutations of a set A, there is a sequence of equal length which starts with a permutation of A would not only be of mathematical interest, but it would also close the gap between Corollaries 1 and 2 as stated in Remark 1.

As a final note, recall that in optimistic two-party contract signing protocols, four messages in the main protocol are necessary and sufficient to achieve fairness [23]. The conjectured lower bound for optimistic multi-party contract signing protocols with $\alpha > 2$ signers is α^2+1. It is interesting that this off-by-one connection is paralleled in the related combinatorial problem. The shortest sequence containing all permutations of two elements has length three, while the conjectured formula for the shortest sequence of $\alpha > 2$ elements is $\alpha^2 - 2\alpha + 4$. This formula is known to hold true for $2 < \alpha < 8$.

References

[1] L. Adleman, "Short permutation strings," *Discrete Math.*, vol. 10, pp. 197–200, 1974.

[2] N. Asokan, "Fairness in electronic commerce," PhD Thesis, University of Waterloo, 1998.

[3] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for multi-party fair exchange," IBM Research, Research Report RZ 2892 (90840), Dec. 1996. [Online]. Available: citeseer.ist.psu.edu/article/asokan98optimistic.html

[4] ——, "Optimistic protocols for fair exchange," in *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 1997, pp. 7–17.

[5] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 593–610, 2000.

[6] B. Baum-Waidner, "Optimistic asynchronous multi-party contract signing with reduced number of rounds," in *Automata, Languages and Programming — ICALP 2001*, ser. LNCS, F. Orejas, P. G. Spirakis, and J. van Leeuwen, Eds., vol. 2076. Crete, Greece: Springer-Verlag, July 2001, pp. 898–911. [Online]. Available: citeseer.ist.psu.edu/article/baum-waidner01optimistic.html

[7] B. Baum-Waidner and M. Waidner, "Round-optimal and abuse free optimistic multi-party contract signing," in *Automata, Languages and Programming — ICALP 2000*, ser. LNCS, U. Montanari, J. D. P. Rolim, and E. Welzl, Eds., vol. 1853. Geneva, Switzerland: Springer-Verlag, July 2000, pp. 524–535.

[8] M. Ben-Or, E. Pavlov, and V. Vaikuntanathan, "Byzantine agreement in the full-information model in o(log n) rounds," in *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2006, pp. 179–186.

[9] R. Chadha, M. Kanovich, and A. Scedrov, "Inductive methods and contract-signing protocols," in *CCS '01*. ACM, 2001, pp. 176–185.

[10] R. Chadha, S. Kremer, and A. Scedrov, "Formal analysis of multi-party contract signing," in *CSFW '04*. Washington, DC, USA: IEEE Computer Society, 2004, p. 266.

[11] K. M. Chandy and J. Misra, "How processes learn," in *PODC '85*. ACM, 1985, pp. 204–214.

[12] V. Chvatal, D. A. Klarner, and D. E. Knuth, "Selected combinatorial research problems," Stanford University, Department of Computer Science, Stanford, CA, USA, Tech. Rep. STAN-CS-72-292, June 1972, http://infolab.stanford.edu/pub/cstr/reports/cs/tr/72/292/CS-TR-72-292.pdf.

[13] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. IT-29, no. 12, pp. 198–208, Mar. 1983.

[14] S. Even and Y. Yacobi, “Relations among public key signature systems,” Computer Science Dept., Technion, Haifa, Isreal, Tech. Rep. 175, March 1980.

[15] M. Franklin, Z. Galil, and M. Yung, “An overview of secure distributed computing,” Department of Computer Science, Columbia University, Tech. Rep. TR CUCS-008-92, March 1992.

[16] G. Galbiati and F. Preparata, “On permutation-embedding sequences,” *SIAM Journal on Applied Mathematics*, vol. 30, no. 3, pp. 421–423, May 1976.

[17] J. Garay, M. Jakobsson, and P. MacKenzie, “Abuse-free optimistic contract signing,” in *Advances in Cryptology – CRYPTO’99*, ser. LNCS, M. J. Wiener, Ed., vol. 1666. Santa Barbara, California, USA: Springer-Verlag, Aug. 1999, pp. 449–466.

[18] J. A. Garay and P. D. MacKenzie, “Abuse-free multi-party contract signing,” in *International Symposium on Distributed Computing*, 1999, pp. 151–165. [Online]. Available: citeseer.ist.psu.edu/article/garay99abusefree.html

[19] P. Koutas and T. Hu, “Shortest string containing all permutations,” *Discrete Math.*, vol. 11, pp. 125–132, 1975.

[20] N. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[21] A. Mukhamedov and M. D. Ryan, “Fair multi-party contract signing using private contract signatures,” *Inf. Comput.*, vol. 206, no. 2-4, pp. 272–290, 2008. [Online]. Available: ftp://ftp.cs.bham.ac.uk/pub/authors/M.D.Ryan/06-contract-IC-review.pdf

[22] M. C. Newey, “Notes on a problem involving permutations as subsequences,” Stanford University, Department of Computer Science, Stanford, CA, USA, Tech. Rep. STAN-CS-73-340, March 1973, http://infolab.stanford.edu/pub/cstr/reports/cs/tr/73/340/CS-TR-73-340.pdf.

[23] B. Pfitzmann, M. Schunter, and M. Waidner, “Optimal efficiency of optimistic contract signing,” in *PODC ’98*. ACM Press, 1998, pp. 113–122, extended version as technical report RZ 2994 (#93040), IBM Zürich Research Lab, Feb. 1998.

[24] A. Russell, M. E. Saks, and D. Zuckerman, “Lower bounds for leader election and collective coin-flipping in the perfect information model,” *SIAM J. Comput.*, vol. 31, no. 6, pp. 1645–1662, 2002.

[25] N. Stenning, “A data transfer protocol,” *Computer Networks*, vol. 1, no. 2, pp. 99–110, 1976.