Constructing Optimistic Multi-party Contract Signing Protocols

Barbara Kordy and Saša Radomirović
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
Luxembourg
barbara.kordy@uni.lu, sasa.radomirovic@inf.ethz.ch

***Abstract*—We give an explicit, general construction for optimistic multi-party contract signing protocols. Our construction converts a sequence over any finite set of signers into a protocol specification for the signers. The inevitable trusted third party's role specification and computations are independent of the signer's role specification. This permits a wide variety of protocols to be handled equally by the trusted third party.**

We give tight conditions under which the resulting protocols satisfy fairness and timeliness. We provide examples of several classes of protocols and we discuss lower bounds for the complexity of fair protocols, both in terms of bandwidth and minimum number of messages.

Our results highlight the connection between optimistic fair contract signing protocols and the combinatorial problem of constructing sequences which contain all permutations of a set as subsequences. This connection is stronger than was previously realized.

I. Introduction

Alice would like to take a trip around the world, but she only has a limited budget. She needs to book hotels, transportation, and event tickets with a long list of independent companies. The cheapest offers are non-refundable and are available for a limited time only. Thus, once Alice has collected all offers, she would like to either book all of them at once or none of them at all.

A solution for Alice is to sign one single contract with all companies she would like to buy a service from. She could do this by carrying out a multi-party contract signing (MPCS) protocol. The MPCS protocol would need to be *fair* to ensure that either all signing parties eventually obtain a signed contract or none of the parties do. This would guarantee that either all or none of Alice's bookings are made. The protocol would also have to satisfy a *timeliness* property to prevent the case where Alice or any other signing partner is left in limbo waiting endlessly for the other parties to sign the contract. Finally, the signing partners might be interested in an *abuse-freeness* property which would ensure that Alice cannot bargain with competing companies by proving that she has the choice of canceling the offers or obtaining them by signing the contract.

A practical solution to a scenario such as the one depicted above is provided by the class of deterministic, asynchronous MPCS protocols. As opposed to randomized protocols, this class avoids the requirement of approximately equal computational powers of signers. This class also assumes a more realistic network model than protocols relying on some degree of synchrony in the communication. Deterministic, asynchronous MPCS protocols do, however, require a trusted third party (TTP). We therefore find the subclass of *optimistic* MPCS protocols to be the most useful one, because protocols in this class only involve the TTP when a failure occurs during the course of the signing protocol. Since it is very likely that Alice and the companies would need to have digital credentials registered with an authority in order to execute MPCS protocols, we may assume that such an authority could also play the role of the TTP.

Thus, we consider in this paper the class of deterministic, asynchronous, and optimistic MPCS protocols. Such protocols consist in general of two sub-protocols — the *main* sub-protocol executed by the signers and not involving the TTP, and a *resolve* sub-protocol to be called only in case of a failure in the main sub-protocol and to be executed between a signer and the TTP.

Existing fair MPCS protocols of all flavors have a rigid protocol structure. They consist of a number of rounds in which the signers exchange *promises* to sign a pre-agreed contract, followed by a round in which the actual signatures on the contract are exchanged. These protocols are nearly symmetric in the computation and communication complexity of signer roles, regardless of the signers' computational resources and communication bandwidth.

In this paper, we show how to construct fair MPCS protocols which can be adapted to specific conditions regarding bandwidth and computational resources of individual signers. A salient feature of our construction is that the TTP's protocol role and decision procedure remains the same, regardless of the number of signers or shape of the constructed protocol.

In particular, we show how to construct fair MPCS protocols with an arbitrary number of signers in which a particular signer needs to be active only twice in the protocol: once to send a promise to every signer and the other time to send a signature to every signer. This flexibility of protocol structure and ease of protocol generation is interesting when MPCS protocols are executed by signers with vastly different resources, such as dedicated, commercial signing servers and smart phones.

The input to our construction algorithm is a *signing sequence* [1], that is, a sequence indicating the order in which the signers become active in the protocol execution. By feeding the construction algorithm with signing sequences which satisfy a simple combinatorial property, we obtain fair MPCS protocols.

Contribution: We present a protocol compiler (Algorithm 1) to convert a signing sequence into an MPCS protocol. We prove that fair signing sequences produce fair MPCS protocols (Theorem 3) and we give a simple decision procedure to verify fairness of signing sequences (Theorem 1). Our achievements are generalizations and converses of the results of [1] as discussed in related work below.

We explain how fair signing sequences and consequently fair MPCS protocols can be constructed and give explicit examples. We discuss the complexity of the resulting protocols and pose the problem of finding fair MPCS protocols requiring minimum overall communication bandwidth.

We believe that our construction produces abuse-free protocols, but we do not analyze our protocols with respect to that property in this paper.

Finally, we consider our proofs of fairness to be an interesting starting point for employing a theorem proving tool to automatically prove fairness of similar protocol constructions with an arbitrary, finite number of protocol roles.

Related Work: Multi-party contract signing protocols are a particular instance of a fair and secure computation. A detailed review of existing work on fairness in secure computation protocols in general, but excluding optimistic protocols, can be found in Gordon's thesis [2]. Optimistic fair exchange protocols, which include optimistic MPCS protocols, are discussed in Asokan's thesis [3].

The present paper deals with optimistic MPCS protocols in an asynchronous communication model. It builds on a line of work which includes Garay et al. [4] introducing the abuse-freeness property and *private contract signatures*, Mukhamedov and Ryan [5] developing the notion of *abort chaining attacks*, and Mauw et al. [1] relating fairness of MPCS protocols to a combinatorial property of *signing sequences*.

Garay and MacKenzie [6] have designed MPCS protocols which were later shown by Chadha et al. [7], using the model checker Mocha, to not satisfy fairness in case of four or more signers. Chadha et al. revised the resolve subprotocol of Garay and MacKenzie. Mukhamedov and Ryan [5] have shown that the revised version does not satisfy fairness in case of more than five signers and introduce a new optimistic MPCS protocol. They proved fairness for their protocol by hand and used the NuSMV model checker to verify the case of five signers. Zhang et al. [8] have used the model checker Mocha in order to analyze the protocol of Mukhamedov and Ryan for up to five signers. The analysis did not reveal any flaws.

Mauw et al. [1] used the notion of abort chaining to derive a lower bound on the number of messages necessary to achieve fairness in MPCS protocols. It was shown that abort chaining attacks are possible whenever the signing sequence of the protocol satisfies the following combinatorial property. The sequence does not contain all permutations of the set of signers as subsequences when one regards only those subsequences that start at or after the position of the last signer to appear for the first time in the sequence. Signing sequences satisfying this combinatorial property are said to be *unfair*. Thus, it was shown that protocols giving rise to signing sequences which are unfair suffer from abort-chaining attacks.

It was neither shown nor claimed, however, that *fair* signing sequences give rise to fair MPCS protocols. In fact, it is not clear *a priori* whether abort-chaining attacks are the only possible attacks on fairness. The present work proves that this is indeed the case. Thus the present work provides a converse result to [1].

An attempt to construct a fair MPCS protocol from one particular fair signing sequence was already made in [1]. This protocol was verified by Zhang et al. [8] with the model checker Mocha, which revealed that one protocol message was missing a private contract signature and thus the protocol was not fair in spite of giving rise to a fair signing sequence. Zhang et al. have constructed fair three and four-party MPCS protocols from a variety of signing sequences. In [9], Zhang et al. extend their previous work by model checking abuse-freeness properties and giving a procedure for constructing MPCS protocols from signing sequences. Their construction of the main protocol is related to the one presented in this paper, but less general, and their construction of the resolve protocol is different from ours.

An independent family of asynchronous MPCS protocols was developed by Baum-Waidner and Waidner [10] and verified using the model checker Mocha by Chadha et al. [7] for two up to five signers. Baum-Waidner [11] improved the complexity of the protocol for the case when fewer than half of the participating signers are dishonest.

Paper structure: We introduce the notation used in this paper, our assumptions and background information on MPCS protocols, private contract signatures, and signing sequences in Section II. We describe how to generate an MPCS protocol from a signing sequence in Section III and prove that fair signing sequences produce fair MPCS protocols in Section IV. We illustrate our results by applying the approach described in this paper to known and novel MPCS protocols in Section V. The complexity of the protocols obtained using our approach is analyzed in Section VI. We briefly discuss in Section VII how to generalize our construction to the case where fewer dishonest signers are tolerated in exchange for shorter protocols.

II. Preliminaries

A. Notation and Assumptions

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ be the set of positive integers. We write $[n]$ for the set $\{i \in \mathbb{N} \mid i \leq n\}$.

Let A be a finite set. A finite *sequence* σ over A of length $n \in \mathbb{N}$ is a function $\sigma\colon [n] \to A$. We will write $|\sigma|$ for the length of σ and we will denote the elements in the image of σ by σ_i instead of $\sigma(i)$. To list all elements of σ, we will write $\sigma = (\sigma_1, \ldots, \sigma_n)$, where $n = |\sigma|$. Furthermore, we will write ϵ for the empty sequence, that is the sequence with no elements, and define the length of the empty sequence to be 0. If σ^1, σ^2, …, σ^j are sequences over A, we will write $(\sigma^1, \ldots, \sigma^j)$ for the concatenation of the sequences. We denote by A^* the set of all finite sequences over A.

A sequence ρ is called a *subsequence* of σ, if there is a strictly increasing function $f\colon [|\rho|] \to [|\sigma|]$, such that $\rho_i = \sigma_{f(i)}$. Thus, ρ is a subsequence of $\sigma = (\sigma_1, \ldots, \sigma_n)$ if ρ can be obtained by erasing zero or more symbols from σ.

If X is a finite subset of $\mathbb{N}$, we write $\max X$ and $\min X$, respectively, for the largest and smallest element of X. We define $\max \emptyset = 0$ and $\min \emptyset = \infty$.

We assume that the communication between signers is asynchronous and messages can get lost or be delayed arbitrary long. For ease of exposition, we assume confidential and authentic channels between all protocol participants. The communication channels between signers and the TTP are furthermore assumed to be *resilient*, which means that the messages sent over these channels are guaranteed to be delivered eventually and without modifications.

B. Optimistic Contract Signing Protocols

The goal of a contract-signing protocol is for all signers to issue a *universally verifiable signature* on a pre-agreed contract and for every signer to obtain a *fully signed contract*. A fully signed contract is a set consisting of every signer's universally verifiable signature.

An *optimistic* contract signing protocol has the property that in case of a failure or dispute in the course of the protocol execution, a TTP can be contacted to recover from the failure or resolve the dispute. Optimistic contract signing protocols therefore typically consist of two subprotocols. The *main* protocol is to be executed by the signers only, while the *resolve* protocol is used to contact the TTP.

In this paper, a signer is said to be *honest* if he follows the protocol specification faithfully and quits the main protocol when executing the resolve protocol.

A common structure for the main contract signing protocol is for the signers to exchange *promises* to sign the contract. Once sufficiently many promises have been exchanged, the signers proceed to send universally verifiable signatures to each other. In case of dispute, the promises received by signers serve as evidence for the TTP to decide on how to resolve the dispute.

Contract-signing protocols are expected to satisfy fairness and timeliness. We define fairness and timeliness of optimistic MPCS protocols as follows. The definitions are based on [5].

Definition 1. *An optimistic MPCS protocol for contract m and finite set of signers A is said to be* fair *for an honest signer $P \in A$, if whenever some signer $Q \in A$, $Q \neq P$, obtains a universally verifiable signature on m from P, then P can obtain a universally verifiable signature on m from R, for all $R \in A$.*

It follows from Definition 1 that if an MPCS protocol over a set of signers A is fair for all signers from A, then either all honest signers obtain (with the TTP's help if necessary) a fully signed contract, or no signer obtains any honest signer's universally verifiable signature on the contract.

Definition 2. *An optimistic MPCS protocol is said to satisfy* timeliness*, if each signer has a recourse to stop endless waiting for expected messages.*

In optimistic protocols and with the assumption stated in Section II-A, timeliness can be achieved by being able to contact the TTP at any time using the resolve protocol and by requiring the TTP to immediately respond to requests from signers.

A further desirable property for MPCS protocols is abuse-freeness which was introduced in [4]. We give the formal definition of [6].

Definition 3. *An optimistic MPCS protocol is said to be* abuse-free*, if it is impossible for any set of signers at any point in the protocol to be able to prove to an outside party that they have the power to terminate or successfully complete the contract signing.*

In order to guarantee abuse freeness in an MPCS protocol, a cryptographic primitive called *private contract signature* was developed [4]. A private contract signature is a type of digital signature which can be verified by a designated verifier and a designated TTP and converted into a universally verifiable signature by the signer or the TTP. Although private contract signatures are not necessary for our construction of fair MPCS protocols, we use them in order to be able to analyze our solution with respect to the abuse-freeness property in future work.

Let P, Q be signers and T be the trusted third party. The following definition recalls the main features of private contract signatures. It uses notation from [5]. For a formal definition of private contract signatures, we refer to [4].

Definition 4. *A* private contract signature *by P for Q on text t with respect to T, denoted by $PCS_P(t, Q, T)$, is a cryptographic object with the following properties:*

1) *$PCS_P(t, Q, T)$ can only be created by P. Signer Q can simulate the creation of $PCS_P(t, Q, T)$.*

2) *T, P and Q, but no one else, can tell the difference between the versions created by P or Q.*
3) *T and P, and no one else, can convert $PCS_P(t, Q, T)$ into a universally verifiable signature, denoted by $S_P(t)$.*

The private contract signature $PCS_P(t, Q, T)$ is used by P to promise to Q to sign the text t. Upon receiving $PCS_P(t, Q, T)$, Q can convince himself that P has promised to sign t. Q cannot, however, use this promise to convince anybody else about this fact. Moreover, P and Q know in advance that T (as well as P) is able to convert $PCS_P(t, Q, T)$ into a universally verifiable signature $S_P(t)$.

C. Signing sequences

Signing sequences were introduced in [1] as a simplified representation of MPCS protocols. A signing sequence indicates the order in which the signers become active in the MPCS protocol execution.

Definition 5. *A finite sequence over a finite set A is said to be* complete over A*, if it contains every permutation of the elements of A as a subsequence.*

For instance, the sequence (a, b, c, a, b, c) is not complete over $A = \{a, b, c\}$, because it lacks the subsequence (c, b, a), while the sequence (a, b, c, a, b, c, a) is complete over A.

Given a sequence σ, a subset $SigSet \subset [|\sigma|]$ will be called a *signing set* for σ. It will be used to indicate the positions in σ in which signers will issue signatures on a contract rather than promises.

The following definitions and theorem are based on [1], but are more general than their counterparts in that the signing set allows for a greater set of sequences to be called signing sequences.

Definition 6. *Let σ be a sequence over a finite set A. Then σ is called a* signing sequence *if there exists a signing set $SigSet$ such that the following conditions are satisfied.*

1) *The prefix of length $|A|$ of σ is a permutation of A: $\{\sigma_1, \ldots, \sigma_{|A|}\} = A$.*
2) *The signing set refers to all elements in A: $\{\sigma_i \mid i \in SigSet\} = A$.*
3) *The signing set satisfies the monotonicity condition $(i \in SigSet \wedge j > i) \Rightarrow (j \in SigSet \vee \sigma_j \neq \sigma_i)$.*

A signing set $SigSet$ satisfying the conditions above is said to be proper.

Example 1. *The sequence (a, b, c, b, a, c) over $\{a, b, c\}$ is a signing sequence. The initial permutation is (a, b, c) and proper signing sets for this sequence are supersets of $\{4, 5, 6\}$.*

Definition 7. *Let σ be a signing sequence over a finite set A of length n with proper signing set $SigSet$. Let $l \leq |A|$ and let $f : [l] \to [n]$ be a function. A subsequence $(\sigma_{f(1)}, \ldots \sigma_{f(l)})$ 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) *$f(1) < |A|$;*
3) *$f(l) \in SigSet$;*
4) *$\forall_p\ \sigma_{f(p)} \notin \bigcup_{f(p)<j<f(p+1)} \{\sigma_j\}$.*

AC subsequences are a translation of abort-chaining attacks [5] to signing sequences.

Definition 8. *A signing sequence σ with proper signing set $SigSet$ which has an AC subsequence is called* unfair*. A signing sequence which is not unfair is called* fair*.*

Example 2. *The signing sequence (a, b, a, b) with signing set $\{3, 4\}$ is fair, since neither of the two possible subsequences (a, b) and (b, a) can satisfy all four conditions of Definition 7.*

The sequence (a, b, a) with signing set $\{2, 3\}$, however, is unfair, as the AC subsequence (a, b) shows.

The three preceding definitions can be specialized to the definitions in [1] by requiring that a signing sequence has length at least $2|A|$ and that the signing set is equal to $SigSet = \{|\sigma| - |A| + 1, \ldots, |\sigma|\}$. In this restricted setting, it was shown in [1] that optimistic contract signing protocols which give rise to an unfair signing sequence do not satisfy fairness. Our generalized setting together with the following theorem allows to easily extend those results to a wider class of protocols.

Our main goal in this work, however, is to show the converse to the result stated above: Every fair signing sequence gives rise to a fair optimistic contract signing protocol with our explicit construction. We will use the following theorem to construct fair signing sequences. A less general version of this theorem was proved in [1].

Theorem 1. *Let σ be a signing sequence over a finite set A with proper signing set $SigSet$. For each $c \in A$, let $l(c) = \min\{j \in SigSet \mid \sigma_j = c\}$.*

The sequence σ with signing set $SigSet$ is fair if and only if for every $c \in A$ the sequence $\sigma^c = (\sigma_{|A|}, \ldots, \sigma_{l(c)})$ is complete over $A \setminus \{c\}$.

Proof: Assume σ is fair. Suppose towards a contradiction that ρ is a permutation of $A \setminus \{c\}$ which is *not* a subsequence of σ^c. Then by Lemma 1 below, ρ can be transformed into an AC subsequence of σ contradicting fairness of σ.

Conversely, assume that for all $c \in A$, σ^c is complete over $A \setminus \{c\}$ and suppose towards a contradiction that $s = (\sigma_{f(1)}, \ldots, \sigma_{f(\eta)})$ is an AC subsequence of σ. It follows that $f(\eta) \geq l(c)$. Let $c = \sigma_{f(\eta)}$. We may assume without loss of generality that $f(2) \geq |A|$ and $f(\eta - 1) < l(c)$ (else we would consider a shorter AC subsequence). Since σ^c contains *all* permutations of $A \setminus \{c\}$ as a subsequence and c is the last element of σ^c, it must contain s as a subsequence.

Thus, σ must contain $(\sigma_{g(1)}, \ldots, \sigma_{g(\eta)})$ as a subsequence, where g is an increasing function such that $g(1) \geq |A|$ and $g(\eta) \leq l(c)$.

By condition 4 of Definition 7, it follows that $g(i) \leq f(i)$ or $g(i) > f(i+1)$ for $i = 1, \ldots, \eta - 1$. However, since $(\sigma_{g(1)}, \ldots, \sigma_{g(\eta)})$ is a subsequence of σ^c, it follows that $g(1) \geq |A| > f(1)$, thus $g(1) > f(2)$. Since $g(i+1) > g(i)$, it follows inductively that $g(i) > f(i+1)$ for $i = 1, \ldots, \eta-1$. This implies that $g(\eta) > g(\eta-1) > f(\eta)$ which is a contradiction, since it implies $g(\eta) > f(\eta) \geq l(c)$, but $g(\eta) \leq l(c)$. ∎

The following lemma is used in the proof of Theorem 1.

Lemma 1. *A signing sequence σ has an AC subsequence if there is a permutation $\rho = (\rho_1, \ldots, \rho_{|A|})$ of A such that ρ is not a subsequence of σ^c for $c = \rho_{|A|}$ and σ^c as defined in Theorem 1.*

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

Let $f(|A|) = l(c)$, thus $\rho_{|A|} = \sigma_{f(|A|)} = c$.

Consider the longest suffix $(\rho_j, \rho_{j+1}, \ldots, \rho_{|A|})$ of ρ which is a subsequence of σ^c. (Since ρ itself is not a subsequence of σ^c, it follows that $j > 1$.)

Let $f(j), \ldots, f(|A|)$ be an increasing sequence such that for all i with $j \leq i < |A|$

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

Such a sequence exists, because $(\rho_j, \rho_{j+1}, \ldots, \rho_{|A|})$ is a subsequence of σ^c.

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

We show that $(\sigma_{f(j-1)}, \ldots, \sigma_{f(|A|)})$ is an AC sequence of σ by verifying all conditions of the Definition 7:

- Condition 1 is satisfied since ρ is a permutation.
- Condition 2 is satisfied since $f(j-1) < |A|$.
- Condition 3 is satisfied since $f(|A|) = l(c)$.
- Condition 4 is satisfied by equation (1).

∎

III. Protocol Compiler

In this section we present a protocol compiler, which for every signing sequence over a set of signers A produces an MPCS protocol consisting of a main protocol to be executed by signers from A and a resolve protocol for the trusted third party. The construction of the main protocol is presented is Section III-A and the resolve protocol is explained in Section III-B.

Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ be a sequence over a set A. We define the *leftmost permutation indexes* of σ, denoted by $\mathrm{lmpi}(\sigma)$, to be the set of positions where distinct elements appear for the first time in σ. Formally, $\mathrm{lmpi}\colon A^* \to \mathcal{P}(\mathbb{N})$ is defined recursively as follows. Set $\mathrm{lmpi}(\epsilon) = \emptyset$. Then

$$\mathrm{lmpi}(\sigma_1, \ldots, \sigma_l) = \begin{cases} \mathrm{lmpi}(\sigma_1, \ldots, \sigma_{l-1}) & \text{if } \sigma_l \in \{\sigma_1, \ldots, \sigma_{l-1}\} \\ \mathrm{lmpi}(\sigma_1, \ldots, \sigma_{l-1}) \cup \{l\} & \text{else.} \end{cases}$$

Furthermore, we denote by $\bar{\sigma}$ the reverse sequence of σ, i.e. $\bar{\sigma} = (\sigma_n, \sigma_{n-1}, \ldots, \sigma_1)$. Note that $(\bar{\sigma})_i = \sigma_{n-i+1}$. We define the *rightmost permutation indexes* of σ by $\mathrm{rmpi}(\sigma) = \{i \in \mathbb{N} \mid |\sigma| - i + 1 \in \mathrm{lmpi}(\bar{\sigma})\}$. The rightmost permutation indexes indicate the position of last appearance of every element in σ.

Example 3. *Let $\sigma = (a, b, a, c, a)$ be a sequence over $A = \{a, b, c\}$. We have $\mathrm{lmpi}(a, b, a, c, a) = \{1, 2, 4\}$, as a, b and c appear in σ for the first time in positions $1, 2$ and 4. Furthermore, $\bar{\sigma} = (a, c, a, b, a)$ and $\mathrm{rmpi}(a, b, a, c, a) = \{2, 4, 5\}$.*

We use $\mathrm{prev}_\sigma(i)$ and $\mathrm{next}_\sigma(i)$ to refer, respectively, to the previous and subsequent position of an element σ_i in σ. Let $i \in [|\sigma|]$. We define

$$\begin{aligned} \mathcal{B}_\sigma(i) &= \{j \in [|\sigma|] \mid j < i, \sigma_j = \sigma_i\} \\ \mathrm{prev}_\sigma(i) &= \begin{cases} \max \mathcal{B}_\sigma(i) & \text{if } \mathcal{B}_\sigma(i) \neq \emptyset, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Analogously,

$$\begin{aligned} \mathcal{A}_\sigma(i) &= \{j \in [|\sigma|] \mid j > i, \sigma_j = \sigma_i\} \\ \mathrm{next}_\sigma(i) &= \begin{cases} \min \mathcal{A}_\sigma(i) & \text{if } \mathcal{A}_\sigma(i) \neq \emptyset, \\ |\sigma| + 1 & \text{else.} \end{cases} \end{aligned}$$

For ease of reading, we will leave out the subscript σ in prev_σ and next_σ whenever there is no confusion over which sequence the functions apply to.

In this paper we use m to denote a contract and we assume that m contains the contract text, the set A of involved signers, the corresponding signing sequence σ, and signing set $SigSet$.

A. Main protocol

Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ be a signing sequence over a finite set A of signers. We present an algorithm which produces an MPCS protocol specification from such a sequence. The protocol specification will consist of n *steps*, where the i-th step of the protocol corresponds to the actions specified for the signer σ_i in the signing sequence.

The simplest manner, in which a sequence could be turned into a signing protocol, is to require that in the i-th step, a signer σ_i waits until he receives a message from every signer which has been active prior to the i-th step in the protocol. Then σ_i sends a message to every other signer participating in the protocol. This would lead to redundancies, since between two appearances of a particular signer, there could

be multiple appearances of other signers. Thus, to reduce the communication complexity, we will only require that each signer receives a message from the *last* appearance of all other signers. This is illustrated in Figure 1. The figure shows the messages expected to be received and the messages to be sent by a signer in a particular sequence.

Figure 1. Messages to be received (incoming arrows) and sent (outgoing arrows) for signer d in two subsequent appearances of d.

Formally, we construct a main MPCS protocol specification for a signing sequence which is assumed to be given as part of a contract m as shown in Algorithm 1. The algorithm loops through every element in the signing sequence. Within a loop, a particular signer σ_i's receive and send actions are specified. First the receiving of promises and signatures by the signer is specified and then the sending of promises and signatures. If this is the last appearance of the signer in the protocol, it is ensured that the signer sends out signatures to all other signers to which it has not sent such a signature yet. Then it is ensured that it waits for all signatures that it has not received up to that point.

The sending of messages is denoted by the instruction $send_P(Q,t)$, where P is the sender, Q the recipient and t the message to be sent. The waiting to receive a message is denoted by the instruction $receive_P(Q,t)$, where P is the recipient, Q the sender and t the message to be received. The instructions for sending and receiving messages do not commute. Since the algorithm first prints out *receive* instructions and then *send* instructions, it follows that in each step i, no promise is sent by an honest signer σ_i to another signer σ_j until all promises that are awaited by σ_i in that step are received. To prevent endless waiting, a signer σ_i has an alternative $Res(m, \mathrm{prev}(i))$ to the *receive* instruction. The $Res(m, \mathrm{prev}(i))$ protocol allows signer σ_i to contact the trusted third party in case he has not received all required messages. The $Res(m,i)$ protocol is formally specified in Section III-B. In Algorithm 1, we use the symbol '$+$' to denote alternative branching.

Every promise sent is annotated with the protocol step in which it was sent. This annotation is denoted by (m,i), where m is the contract and i is the protocol step number. This annotation is analogous to the *promise level* of previous works [5], [6].

The set $SigSet$ is the set of positions in σ in which signers will send their universally verifiable signatures to other signers. This set needs to be proper (Definition 6) which ensures that all signers send a signature and that once a signer has issued a signature, it will continue to issue signatures, rather than promises. We will typically set $SigSet = \mathrm{rmpi}(\sigma)$, that is, each signer sends signatures in its last appearance in the signing sequence. We refer to Remark 3 in Section V for a class of protocols which requires a larger $SigSet$.

The construction of the formal specification of the protocol is given in Algorithm 1. The input to this algorithm consists of contract m from which it is possible to extract the set of signers A, the signing sequence σ and the signing set $SigSet$. The algorithm outputs a formal specification of our MPCS protocol.

Algorithm 1: Main protocol compiler

input : m
output: $Main(m)$

1 **for** $i \in [|\sigma|]$ **do**
2 **for** $j' \in \mathrm{rmpi}(\sigma_{\mathrm{prev}(i)+1}, \ldots, \sigma_{i-1})$ **do**
3 $j := \mathrm{prev}(i) + j'$;
4 **if** $j \notin SigSet$ **then**
5 **print** "$(receive_{\sigma_i}(\sigma_j, PCS_{\sigma_j}((m,j),\sigma_i,T))+ Res(m,\mathrm{prev}(i)))$";
6 **else**
7 **print** "$(receive_{\sigma_i}(\sigma_j, S_{\sigma_j}(m))+ Res(m,\mathrm{prev}(i)))$";
8 **for** $j' \in \mathrm{lmpi}(\sigma_{i+1}, \ldots, \sigma_{\mathrm{next}(i)-1})$ **do**
9 $j := i + j'$;
10 **if** $i \notin SigSet$ **then**
11 **print** "$send_{\sigma_i}(\sigma_j, PCS_{\sigma_i}((m,i),\sigma_j,T))$";
12 **else**
13 **print** "$send_{\sigma_i}(\sigma_j, S_{\sigma_i}(m))$";
14 **if** $\mathrm{next}(i) = |\sigma| + 1$ **then**
15 $i_0 := \min\{j \in SigSet \mid \sigma_i = \sigma_j\}$;
16 **for** $P \in A \setminus \{\sigma_{i_0}, \ldots, \sigma_{|\sigma|}\}$ **do**
17 **print** "$send_{\sigma_i}(P, S_{\sigma_i}(m))$";
18 **for** $Q \in A \setminus \{\sigma_j \mid j \in SigSet \wedge j \leq i\}$ **do**
19 **print** "$(receive_{\sigma_i}(Q, S_Q(m))+Res(m,i))$";

Remark 1. Note that the specification for an individual signer P's protocol role can be obtained from Algorithm 1 by changing the set $[|\sigma|]$ in line 1 to the set of positions in which signer P appears in the signing sequence, i.e., the set $\{l \in [|\sigma|] \mid \sigma_l = P\}$.

Example 4. *The signing sequence* $\sigma = (a,b,c,b,\ a,b,c,b,\ a,b,c)$ *with* $SigSet = \mathrm{rmpi}(\sigma)$ *generates a three-signer MPCS protocol of Mukhamedov and Ryan [5] and leads to the protocol shown in Figure 2. The arrows' tails and heads correspond, respectively, to the* send *and* receive *instructions generated by Algorithm 1. The top arrow connecting the vertical lines under* a *and* b *corresponds, for*

instance, to the instructions $send_a(b, PCS_a((m,1),b,T))$ *and* $receive_b(a, PCS_a((m,1),b,T))$. *If we change line 1 in the algorithm according to the preceding remark, we obtain the* receive *and* send *instructions which can be read along the vertical lines in Figure 2.*

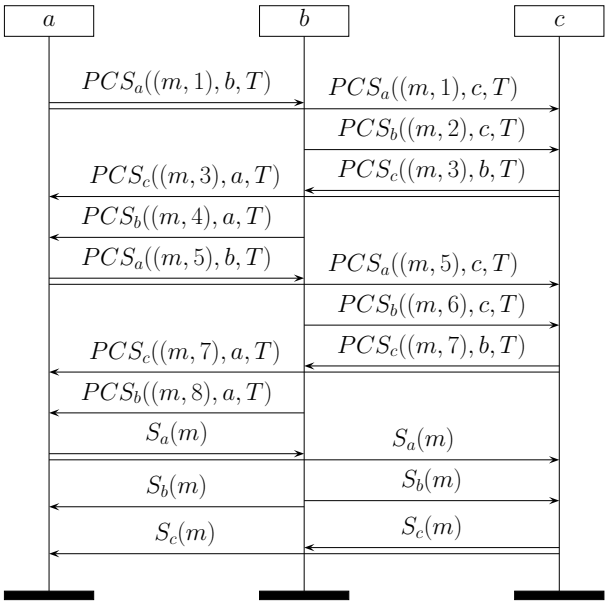

Figure 2. Protocol generated by Algorithm 1 from signing sequence $(a,b,c,b,\ a,b,c,b,\ a,b,c)$

1) Correctness: We show that the compiled main protocol is well-defined and that when all signers are honest, every signer will receive a fully signed contract. This property is independent of whether the signing sequence is fair or not.

The case where one or more signers deviate from or abort the protocol is correct in the sense that every honest signer has the option to run the resolve protocol as an alternative to the *receive* instruction after a specific timeout.

Lemma 2. *The* $Main(m)$ *protocol is well-defined. That is, there is a one-to-one correspondence between* $send_Q(P,t)$ *and* $receive_P(Q,t)$ *instructions and every such* send *instruction precedes the corresponding* receive *instruction.*

Proof: We consider the protocol specification generated by the two inner **for** loops of Algorithm 1 separately from the specification produced by the remainder of the algorithm.

1) The protocol specification generated in lines 2 through 13 is well defined.
   We first show inductively that the *send* instructions precede the *receive* instructions. Signer σ_1 starts the protocol by sending promises to other signers. In particular, σ_1 does not wait to receive any messages, since the condition in line 2 of the protocol specification in Algorithm 1 specifies an empty sequence and thus the resulting set is empty.
   Signer σ_i, for $1 < i < |\sigma|+1$, only waits for messages from signers σ_j where $j < i$.
   It therefore remains to be shown that there is a one-to-one correspondence between *send* and *receive* instructions. To show that for every *receive* instruction for σ_i in line 5 (line 7, respectively), there is a corresponding *send* instruction for σ_j in line 11 (line 13, respectively), amounts to showing that for every
   $$j \in \left\{\mathrm{prev}(i) + j' \mid j' \in \mathrm{rmpi}(\sigma_{\mathrm{prev}(i)+1}, \ldots, \sigma_{i-1})\right\} \tag{2}$$
   there exists
   $$i' \in \mathrm{lmpi}(\sigma_{j+1}, \ldots, \sigma_{\mathrm{next}(j)-1})$$
   such that $i = j + i'$.
   This is true, since by (2) and definition of rmpi we have $\mathrm{prev}(i) < j < i < \mathrm{next}(j)$. Thus $i \in \left\{j + i' \mid i' \in \mathrm{lmpi}(\sigma_{j+1}, \ldots, \sigma_{\mathrm{next}(j)-1})\right\}$.
   The one-to-one correspondence now follows from the fact that no two *send* or *receive* instructions are identical and that the number of *send* instructions printed equals the number of *receive* instructions printed. The latter follows from the facts that rmpi and lmpi always have the same number of elements when applied to the same sequence and that the two inner **for** loops range over the same non-empty sequences.
2) To show that the protocol specification generated in lines 14–19 is also well defined, we need to prove that for every *receive* instruction in line 19 there exists a corresponding *send* instruction and for every *send* instruction in line 17 there is a corresponding *receive* instruction.
   Let i be an index such that $\mathrm{next}(i) = |\sigma| + 1$ and let $Q \in A \setminus \{\sigma_j \mid j \in SigSet \wedge j \leq i\}$. According to specification in line 19, signer σ_i waits for a signature from Q. We will show that a corresponding *send* by Q is specified in line 17. Since $Q \in A \setminus \{\sigma_j \mid j \in SigSet \wedge j \leq i\}$, either
   a) every position l, such that $l \in SigSet$ and $\sigma_l = Q$, is grater than i, or
   b) $\{l \mid \sigma_l = Q\} \cap SigSet = \emptyset$, i.e., none of the indexes in $SigSet$ corresponds to signer Q.

   Let $l_0 = \min\{j \in SigSet \mid Q = \sigma_j\}$. If case 2a holds, then $i < l_0$. Together with the fact that $\mathrm{next}(i) = |\sigma| + 1$ this means that $\sigma_i \in A \setminus \{\sigma_{l_0}, \ldots, \sigma_{|\sigma|}\}$. If case 2b holds, then $A \setminus \{\sigma_{l_0}, \ldots, \sigma_{|\sigma|}\} = A$. In both cases, according to lines 16 and 17, there exists a *send* instruction $send_Q(\sigma_i, S_Q(m))$.
   We now show that for every *send* instruction in line 17 there exists a corresponding *receive* instruction in line 19. Recall that $i_0 := \min\{j \in SigSet \mid \sigma_i = \sigma_j\}$

in line 15. We still consider i, such that $\mathrm{next}(i) = |\sigma|+1$ and we fix $P \in A \setminus \{\sigma_{i_0}, \ldots, \sigma_{|\sigma|}\}$. According to line 17, signer σ_i sends his signature to signer P. To find a corresponding *receive* instruction for P, we consider the last appearance of P in σ and denote the corresponding position by l so that $\sigma_l = P$ and $\mathrm{next}(l) = |\sigma|+1$. According to lines 18 and 19, it is sufficient to show that $\sigma_i \in A \setminus \{\sigma_j \mid j \in SigSet \wedge j \leq l\}$. Since, $P \in A \setminus \{\sigma_{i_0}, \ldots, \sigma_{|\sigma|}\}$, we know that $l < i_0 \leq i$. Therefore, $\sigma_i \notin \{\sigma_j \mid j \in SigSet \wedge j \leq l\}$ which is equivalent to $\sigma_i \in A \setminus \{\sigma_j \mid j \in SigSet \wedge j \leq l\}$.

■

Theorem 2. *Let σ be a signing sequence over a finite set A with proper signing set $SigSet$. If all signers are honest then every signer $P \in A$ receives from every signer $Q \in A$, $Q \neq P$, a signature $S_Q(m)$.*

Proof: Since the protocol is well-defined by Lemma 2, it remains to be shown that every signer receives a fully signed contract. Since $SigSet$ is proper, every signer sends a signature. By lines 16 and 17, every signer who has not received a signature from σ_i by the last appearance of σ_i, is sent a signature. ■

B. Resolve Protocol

Our resolve protocol is a two-message protocol in which a signer who has sent a message in the i-th protocol step submits evidence supporting the fact that the protocol execution has reached the i-th step. The TTP stores evidence submitted by every requesting signer and considers it together with existing evidence and *immediately* sends back an abort token or a signed contract. We denote the TTP by T in the protocol.

Recall our assumptions that the communication channels for this protocol are resilient, confidential, and authentic. Recall further that the contract m is assumed to contain the contract text, the set A of signers, the signing sequence σ over A and corresponding signing set $SigSet$.

For the signers, the resolve protocol depends on the last protocol step, i, in which the signer to execute the resolve protocol has sent a message. We will denote the resolve protocol for this step i by $Res(m, i)$. If a signer has not sent any messages, we set $i = 0$. For the TTP, the resolve protocol is independent of the signing sequence or protocol step a signer is in. It is the TTP's decision procedure which processes the evidence submitted and returns an abort token or a fully signed contract.

$Res(m, i)$ is the protocol whose two role specifications are shown in Figures 3 and 4, where the variables are as follows. Let $p^i_{\sigma_i}$ denote the set of each signer's most recent promise or universally verifiable signature received by signer σ_i until the i-th step of the protocol. Thus

$$\begin{aligned} p^i_{\sigma_i} \quad = \quad & \{PCS_{\sigma_j}((m,j),\sigma_i,T) \mid \\ & \qquad j \in \mathrm{rmpi}(\sigma_1,\ldots,\sigma_i) \setminus SigSet, \sigma_j \neq \sigma_i\} \\ & \cup \quad \{S_{\sigma_j}(m) \mid \\ & \qquad j \in \mathrm{rmpi}(\sigma_1,\ldots,\sigma_i) \cap SigSet, \sigma_j \neq \sigma_i\} \end{aligned}$$

If $i = 0$, then the set $p^i_{\sigma_i}$ is defined to be empty. *History* denotes the following set of signatures:

$$History = p^i_{\sigma_i} \cup \{PCS_{\sigma_i}((m,i),\sigma_i,T)\}. \qquad (3)$$

The private contract signature $PCS_{\sigma_i}((m,i),\sigma_i,T)$ is explicitly added to the *History* set, because the set $p^i_{\sigma_i}$ does not contain any promise of signer σ_i. This additional promise serves several purposes. It allows the TTP to extract the contract m even when $p^i_{\sigma_i}$ is empty. It allows the TTP to extract the position i at which the signer σ_i claims to have received promises or signatures for the last time in the protocol execution. It also allows the TTP to create a universally verifiable signature on behalf of signer σ_i in cases where σ_i is the first signer to contact the TTP.

The term *decision* is either a fully signed contract, i.e. the set $\{S_P(m) \mid P \in A\}$, or an abort token which we will simply denote by "abort" and which we do not assume to have any further meaning.

$$\begin{aligned} & send_{\sigma_i}(T, p^i_{\sigma_i} \cup \{PCS_{\sigma_i}((m,i),\sigma_i,T)\}) \\ & receive_{\sigma_i}(T, decision) \end{aligned}$$

Figure 3. Signer σ_i's role in resolve protocol $Res(m, i)$.

$$\begin{aligned} & receive_T(P, History) \\ & send_T(P, decision_m) \end{aligned}$$

Figure 4. TTP role in resolve protocol $Res(m, i)$.

For each contract m, the TTP maintains a data structure consisting of a set of private contract signatures $Evidence_m$, an index set I_m, a set $Dishonest_m$ of signers considered to be dishonest, and a variable $decision_m$. The sequence σ and the set of signers A corresponding to the main protocol are extracted from m. The set $Evidence_m$ is the union of *History* sets which the TTP receives in the first message of the resolve protocol. The set of indexes I_m is the set of positions i in σ at which signers executed $Res(m, i)$ to contact the TTP. Thus, for $i \in I_m$, the signer σ_i has contacted the TTP claiming to have received the elements of $p^i_{\sigma_i}$ and sent his promises or signatures, but not received all expected messages necessary for his subsequent protocol step, $\mathrm{next}(i)$. The variable $decision_m$ is equal to the abort token or a set of universally verifiable signatures on m, one by each signer, according to the TTP's last decision on whether to issue an abort token or a signed contract.

We will write $decision_m = \perp$ if the TTP has not been contacted by any signer regarding contract m.

The TTP's decision procedure, shown in Algorithm 2, works as follows. When the TTP receives a message from P, the TTP first verifies that the message received is valid and stores the evidence submitted (lines 1 through 8). The remainder of the algorithm concerns the detection of signers who have continued the main protocol execution after executing the resolve protocol. This part of the algorithm is similar to the TTP decision procedure of [5] and works as follows.

- If P has not received a promise from every other signer in the protocol, that is, if the last position in which P has sent a message in the protocol is smaller than the number of signers participating in the protocol (line 9) then the TTP sends back the last decision made. This decision is an "abort" token unless the TTP has been contacted before and decided to send back a signed contract.
- If P has received a promise from every other signer in the protocol (lines 12 and up) the TTP is able to create a signed contract from P's evidence. In this case, the TTP's decision procedure works as follows.
  - If there is a previous decision to send back a signed contract, then the TTP sends back a signed contract to P (line 13).
  - Else the TTP computes the set of dishonest signers by adding to it every signer which has carried out the resolve protocol, but can be seen to have continued the protocol execution (line 17) based on the evidence the TTP has collected.
    - If P is in the set of dishonest signers, an "abort" token is sent to P.
    - If at least one signer other than P is honest (line 21), an "abort" token is sent to P, because the other signer must have received an abort token already.
    - Else P must be the only honest signer that has contacted the TTP until this point in time and therefore it is safe to return a signed contract.

1) Correctness: We show that the resolve protocol is correct in the sense that:

- The protocol $Res(m, i)$ is well-specified. This means that a signer executing $Res(m, i)$ can generate the correct evidence and that the TTP has the ability to verify the correctness of the submitted evidence.
- The TTP can issue signatures whenever his decision is to resolve rather than to abort the signing protocol.
- No honest signer will be placed in the set $Dishonest_m$.

These properties are independent of whether the signing sequence is fair or not.

Lemma 3. *The protocol $Res(m, i)$ satisfies the following properties.*

Algorithm 2: TTP decision procedure

input : $P, History, m, \sigma, A, i$
output: $decision_m$

1 **if** $decision_m = \perp$ **then**
2 $Evidence_m := \emptyset$; $I_m = \emptyset$;
3 $Dishonest_m := \emptyset$; $decision_m :=$ "abort";
4 **if** $P \in Dishonest_m \vee \exists j \in I_m : P = \sigma_j \vee History \neq p^i_{\sigma_i} \cup \{PCS_{\sigma_i}((m, i), P, T)\}$ **then**
5 $Dishonest_m := Dishonest_m \cup \{P\}$;
6 **return** *"abort"*;
7 $I_m := I_m \cup \{i\}$;
8 $Evidence_m := Evidence_m \cup History$;
9 **if** $i < |A|$ **then**
10 **return** $decision_m$;
11 **else**
12 **if** $decision_m \neq$ *"abort"* **then**
13 **return** $decision_m$;
14 **else**
15 $l := \max I_m$;
16 **for** $j \in I_m, j < l$ **do**
17 **if** $\sigma_j \in \{\sigma_{j+1}, \ldots, \sigma_l\}$ **then**
18 $Dishonest_m := Dishonest_m \cup \{\sigma_j\}$;
19 **if** $P \in Dishonest_m$ **then**
20 **return** $decision_m$;
21 **if** $\exists j \in I_m : \sigma_j \notin Dishonest_m \wedge \sigma_j \neq P$ **then**
22 **return** $decision_m$;
23 **else**
24 $decision_m := \{S_Q(m) \mid Q \in A\}$;
25 **return** $decision_m$;

1) *An honest signer executing $Res(m, i)$ has the ability to generate the set History as defined in equation* (3).
2) *The TTP can verify the correctness of the set History.*
3) *The TTP has the ability to generate a fully signed contract whenever the decision procedure requires him to do so.*

Proof:

1) Clearly a signer P can generate $PCS_P((m, i), P, T)$. It remains to show that an honest signer can generate the set $p^i_{\sigma_i}$ after protocol step i. Note first that every index $j \in \text{rmpi}(\sigma_1, \ldots, \sigma_i)$ such that $\sigma_j \neq P = \sigma_i$ is present in one of the sets $\{\text{prev}(i') + j' \mid j' \in \text{rmpi}(\sigma_{prev(i')+1}, \ldots, \sigma_{i'-1})\}$ where $1 \leq i' \leq i$ and $\sigma_{i'} = \sigma_i$. Thus, the elements of $p^i_{\sigma_i}$ are a subset of the messages specified to be obtained by σ_i through the *receive* instructions in the first inner **for** loop of Algorithm 1.
   By the well-specification of the $Main(m)$ protocol

(Lemma 2), each of these messages must have been sent prior to the i-th protocol step. Since an honest signer executes $Res(m, i)$ only after having sent the messages in the i-th protocol step, it follows that P must have received all the necessary messages.

2) The TTP can verify whether the set *History* submitted satisfies equation (3), as follows. Since the set *History* always contains $PCS_P((m, i), P, T)$, the TTP can extract the contract m and the claimed protocol step i from *History*. The signing sequence can be extracted from m. Thus the TTP can verify that $P = \sigma_i$, that is, the signer contacting the TTP is the same as the signer which is supposed to be in step i. Finally, the contents of the set $p^i_{\sigma_i}$ can be verified by property 2 of Definition 4.
3) The TTP changes his decision from "abort" to a fully signed contract only in line 24 of the decision procedure (Algorithm 2). This line is only reached when $i > |A|$, that is when every signer has appeared in the protocol. Thus, the set *History* will contain a promise or signature by every signer participating in the protocol and the TTP will be able to convert promises to universally verifiable signatures by property 3 of Definition 4.

■

The following lemma states that no signer can convince the TTP that the protocol has progressed beyond an honest signer's next protocol step. It will be used to show that honest signers will not be considered to be dishonest by the TTP.

Lemma 4. *Let σ be a signing sequence and $Main(m)$ the corresponding main contract signing protocol. If an honest signer σ_i has not sent any message in the i-th protocol step of $Main(m)$, then any signer claiming to have reached a protocol step $j > i$ will have an incorrect History set.*

Proof: Since $j > i$, there exists $i' \in \mathrm{rmpi}(\sigma_1, \ldots, \sigma_j)$ such that $i' \geq i$ and $\sigma_{i'} = \sigma_i$. Thus the set *History* needs to contain a signature by σ_i at step i'. The existence of such a signature contradicts the fact that σ_i is honest and did not send any message in the i-th step. ■

Lemma 5. *If signer P is honest in $Main(m)$, then for all behaviors of signers $Q \in A$, $Q \neq P$ and every run of the TTP decision procedure, $P \notin Dishonest_m$.*

Proof: A signer is placed in to the set $Dishonest_m$ as a consequence of lines 4 and 17 in the TTP decision procedure.

An honest signer will not be placed in $Dishonest_m$ as a consequence of any of the three conditions in line 4, since $Dishonest_m$ is initially empty, an honest signer by definition quits the protocol execution after contacting the TTP and sends messages as specified in the protocol. Regarding the last of the three conditions, by the protocol specification in Figure 3 and by equation 3, the honest signer sends precisely the set of messages the TTP expects.

An honest signer will not be placed in $Dishonest_m$ as a consequence of the condition in line 17: If an honest signer appears in I_m, then the signer must have executed the resolve protocol $Res(m, i)$ for some $i \in I_m$. This is because the only line modifying the set I_m is line 7, which is executed as a consequence of a signer σ_i contacting the TTP through an authentic channel and presenting a correct *History* set. By definition, an honest signer quits after executing the resolve protocol and by Lemma 4, no other signer can produce a correct *History* set after protocol step i. ■

IV. Proving Fairness and Timeliness

We prove fairness of the protocol constructed using the protocol compiler described in Section III with TTP procedure shown in Algorithm 2 in three steps, split over three lemmas. The following lemma expresses that the TTP decision procedure enforces all but one of the abort chaining conditions of Definition 7. The subsequent lemma shows that if an honest signer sends a signature, then the remaining condition is satisfied, too. This allows us to directly link fairness of the contract signing protocol in this case to the fairness property of the signing sequence. The third lemma deals with the case in which an honest signer has received an abort token and states that no other signer will receive a fully signed contract.

The lemmas are combined to prove fairness of the constructed MPCS protocol in the subsequent theorem which also addresses the timeliness property.

Lemma 6. *Let m be a contract and σ the contract's signing sequence of length n. We have the following invariant in the TTP's decision procedure for $\sigma, I_m, decision_m$:*

If $decision_m =$ "abort" and $I_m \neq \emptyset$ then there exist l with $0 < l \leq |I_m|$, $I \subset [n]$, and a bijective, increasing function $f\colon [l] \to I$ with $f(l) = \max I_m$ such that conditions 1, 2, 4 of Definition 7 are satisfied for the subsequence $(\sigma_{f(1)}, \ldots, \sigma_{f(l)})$ of σ.

Proof: To distinguish the data structures before and after a TTP run for contract m, we will write I_m, f, $decision_m$ for the values before the TTP's run and I'_m, f', $decision'_m$ for the values after the run.

Note that $decision_m$ is set to "abort" upon the first run for contract m. Suppose $decision_m =$ "abort". If $|I_m| = 1$, then the TTP decision procedure obviously enforces that the unique element $i \in I_m$ satisfies $i < |A|$. Thus conditions 1, 2, 4 are satisfied for $f(1) = i$.

Suppose that the conditions are satisfied for I_m, f before a TTP run and suppose that $decision'_m =$ "abort" after the TTP run. We need to show that there exists f' such that the conditions remain satisfied. If $I_m = I'_m$, then we set $f' = f$ and there is nothing to be shown. Else, there is a unique

element $i \in I'_m \setminus I_m$ that has been added. If $i \neq \max I'_m$, then the conditions remain satisfied for the function $f' = f$.

Thus suppose $i = \max I'_m$. By line 4 of the decision procedure of the TTP, condition 1 remains satisfied, regardless of choice of function f'. Since $decision'_m =$ "abort", by line 21, there exists $j \in I'_m$, $j \neq i$, such that $\sigma_j \notin \{\sigma_{j+1}, \ldots, \sigma_i\}$. Thus we define $f'(x) = f(x)$ for all $x \in [l]$ such that $f(x) < j$. Let x be the maximal element in $[l]$ such that $f(x) < j$, then define $f'(x+1) = j$ and $f'(x+2) = i$ to obtain a bijective, increasing function $f' \colon [x+2] \to I'_m$.

Since all elements in I_m, which are smaller than or equal to j, are still in the preimage of f', condition 2 remains satisfied.

Condition 4 is satisfied for all elements smaller than j in the preimage of f', because it was satisfied for f. It is satisfied for j by lines 16, 17 and line 21 and vacuously satisfied for i. ∎

Lemma 7. *Let m be a contract with a fair signing sequence σ and a proper signing set $SigSet$. If an honest signer P sends $S_P(m)$ in the main protocol, then P will receive all signatures.*

Proof: Let i be the earliest step in which P has sent a signature. Since σ is fair, it follows that $i > |A|$.

Suppose that P does not receive all signatures. Then P will eventually execute the resolve protocol $Res(m, j)$ for $j \geq i$. Since P is honest, by Lemma 5, $P \notin Dishonest_m$ in Algorithm 2. Since $j \geq i > |A|$ and $P \notin Dishonest_m$, the **if** conditions in lines 9, 12, 19, 21 of the TTP decision procedure either lead to a fully signed contract or to an "abort" token in line 22.

Suppose towards a contradiction that P receives an "abort" token. By line 21 of the TTP decision procedure, there must be another signer $Q \notin Dishonest$ to whom an abort token is issued. Thus, by Lemma 6, there exist $l \in \mathbb{N}$ and an increasing function $f \colon [l] \to I$ such that $f(l) = \max I$. Since signer P has sent a signature, condition 3 of Definition 7 is satisfied. By Lemma 6, the subsequence $(\sigma_{f(1)}, \ldots, \sigma_{f(l)})$ of σ satisfies conditions 1, 2, 4, too. This is a contradiction, since σ is a fair sequence. ∎

The following lemma expresses that if an honest signer receives an abort token before he reaches a step in the signing set, then no signer will reach the signing set. More precisely, any signer contacting the TTP with $Res(m, i)$, $i \in SigSet$ will be considered dishonest. Together with the fact that no honest signer will be considered dishonest by the TTP, it follows that only dishonest signers can pretend to have issued a signature.

Lemma 8. *Suppose σ is a fair sequence over A with proper signing set $SigSet$. Suppose further that an honest signer P receives an abort token in $Res(m, i)$ without having sent a signature $S_P(m)$ to any signer. Then no signer will receive the signature $S_P(m)$.*

Proof: Let i be the protocol step in which P has contacted the TTP. Since P has not sent $S_P(m)$ and P is honest, only the TTP could have generated $S_P(m)$.

Suppose a signer $Q \neq P$ has contacted the TTP in step j. By the TTP decision procedure, if $j < i$, then Q must receive an abort token, since P has received an abort token. If $j > i$, since P is honest, P did not participate in the main protocol after receiving the abort token. Thus, by Lemma 5, $P \notin Dishonest_m$, therefore Q must receive an abort token. ∎

We are now ready to state and prove our main theorem, namely that our construction provides fairness for protocols constructed from fair signing sequences and timeliness for all signing sequences.

Theorem 3. *Let m be a contract with fair signing sequence σ and proper signing set $SigSet$. Then $Main(m)$ as created by Algorithm 1 and $Res(m, i)$ with the decision procedure stated in Algorithm 2 constitute an optimistic contract signing protocol satisfying fairness and timeliness.*

Proof:

- Fairness
  Suppose the contract signing protocol for m is not fair for an honest signer P. By Definition 1, there is a signer $Q \neq P$ who has obtained P's universally verifiable signature $S_P(m)$, but P does not have a fully signed contract. Thus, either P has sent his signature but has not received all signatures or P has contacted the TTP and has received an "abort" token. The former case is impossible by Lemma 7 and the latter case is impossible by Lemma 8.
- Timeliness
  No signer will wait endlessly for another signer, since it has an option to execute the resolve protocol at every step in which it waits for a message by another signer. No signer will wait endlessly for the TTP, since the communication channel between the TTP and the signer is assumed to be resilient and the TTP immediately responds to resolve requests by signers.

∎

V. Constructing Fair Protocols

By Theorems 1 and 3, to obtain a fair MPCS protocol one needs to construct a complete sequence and a proper signing set. We now show how this can be done explicitly and provide examples of known and novel, fair MPCS protocols.

We first reduce the problem to the construction of complete sequences.

Theorem 4. *Let A be a set of signers and τ a complete sequence over A. Let ρ be a permutation of $A \setminus \{\tau_1\}$. Then $\sigma = (\rho, \tau)$ is a fair signing sequence for the signing set $SigSet = \mathrm{rmpi}(\sigma)$.*

Proof: It is easy to see that σ is a signing sequence with the proper signing set $SigSet$.

Let $l(c)$, σ^c be as defined in Theorem 1. Let $i \in SigSet$, $j \in \mathrm{rmpi}(\tau)$ such that $\sigma_i = \tau_j$. Let $c = \sigma_i$. Since $i \in SigSet = \mathrm{rmpi}(\sigma)$, $l(c) = i$. Thus, $\sigma^c = (\tau_1, \ldots, \tau_j)$.

Since τ is a complete sequence over A and $j \in \mathrm{rmpi}(\tau)$, the sequence $\sigma^c = (\tau_1, \ldots, \tau_j)$ is complete over $A \setminus \{\tau_j\}$.

By Theorem 1, σ is a fair sequence. ∎

Thus, to construct a fair protocol, we start with a complete sequence. We then prepend a permutation of elements in A to the complete sequence to obtain a signing sequence σ. We chose the signing set to be $\mathrm{rmpi}(\sigma)$, i.e. the indexes of the last appearance of every element in A. Then we apply the protocol compiler.

Constructions for complete sequences over arbitrary finite sets have been given by Adleman [12], Newey [13], and Mohanty [14]. They give rise to signing sequences of length $k^2 - k + 3$.

MR protocols: Let $A = \{P_1, \ldots, P_k\}$ be a set of signers. Then the sequences $(P_1, \ldots, P_{k-1}, P_k, P_{k-1}, \ldots, P_2)$ concatenated $\lceil k/2 \rceil$ times and extended by $(P_1, \ldots, P_k)$ lead to the protocols proposed by Mukhamedov and Ryan [5]. The $SigSet$ consists of the last k indexes. Fairness of signing sequences obtained by this construction can be shown easily. A sketch of the proof goes as follows. By construction, there are $k-1$ concatenated permutations over A after the initial permutation and before the elements indexed by $SigSet$. Thus, these $k-1$ permutations form a complete sequence over $A \setminus \{c\}$, for every $c \in A$. The number of messages to be sent in MR protocols, which is related to the bandwidth complexity discussed in the next section, was computed by Mukhamedov and Ryan to be $k(k-1)(\lceil k/2 \rceil + 1)$.

MRT protocols: The protocols generated in this paper can be easily modified in such a way that the signers send only one message in each step of the protocol. Instead of sending separate promises (or signatures) to several signers, a signer $P = \sigma_i$ sends all promises (or signatures) to the next signer $Q = \sigma_{i+1}$ in the signing sequence. Signer Q then forwards all of P's promises (or signatures) to the subsequent signer, exchanging the promises (or signatures) intended for Q by promises (or signatures) generated by Q. This leads to a class of protocols which were investigated by Mauw et al. [1]. The authors computed a lower bound in terms of the total number of messages that must be sent in such protocols for the protocols to be fair. We will refer to this type of complexity measure as the *message complexity* of the generated protocol in the next section. The authors constructed one shortest possible protocol for three signers. Zhang et al. [8], [9] give descriptions of more protocols based on shortest signing sequences to which they refer as MRT protocols. We thus *define* an MRT protocol to be the shortest possible fair MPCS protocol in terms of message complexity.

The constructions of Adleman [12] and Newey [13] produce complete sequences of length $k^2 - 2k + 4$. These are known to be the shortest lengths for complete sequences when $2 < k < 8$, thus they lead via Theorem 4 and our protocol compiler to MRT protocols for more than 2 and up to 7 signers. The shortest length for complete sequences when $k \geq 8$ is still open and will be discussed further in Section VI.

Remark 2. It is interesting to note that with our generalization of signing sequences we are able to produce shorter protocols for certain sequences than is possible with the original notion of Mauw et al. used by Zhang et al. The sequences in question are those which neither start nor end with a permutation of the set of signers. An example can be constructed from a sequence first described by Newey [13]:

$$(1,2,3,4,5,1,6,4,3,2,1,5,4,3,\\ 6,1,2,4,5,3,1,6,4,2,3,5,1,4)$$

The sequence shown is a shortest possible sequence over a set of six elements containing permutations of all six elements as subsequences. The more restricted definition of a signing sequence in previous work required, however, that a signing sequence ends with a permutation. This necessitates an additional appearance of signer 6 at the end of the sequence shown above.

Remark 3. Since MRT protocols require promises and signatures to be forwarded rather than sent directly to their destination, it is necessary to specify in which order the last signer's signature is to be forwarded to the other signers. This can be achieved by appending these $|A| - 2$ signers in desired order to a fair signing sequence and *adding* their positions in the resulting signing sequence to the $SigSet$. Thus, the $SigSet$ contains $2|A| - 2$ elements for contract signing protocols with message forwarding.

Minimizing the involvement of certain parties: The involvement of one signer in the protocol can be reduced by increasing the involvement of other parties. This can be useful if certain signers have a significantly reduced bandwidth compared with the other signers. In an extreme case, a fair protocol can be produced in which a signer sends promises in one step and signatures in a further step, independent of the number of other signers. This comes, however, at the cost of almost doubling the number of all other signers' steps.

The construction is as follows.

Theorem 5. *Let $\sigma = (\rho, \tau)$ be a fair sequence, where ρ is a permutation of $A \setminus \{\tau_1\}$. Let $s \notin A$ be a signer which is not in the set A. Then the sequence $\sigma' = (s, \rho, \tau, s, \tau)$ is a fair sequence over $A \cup \{s\}$ for the signing set $SigSet = \mathrm{rmpi}(\sigma')$.*

The proof of the theorem is standard and therefore omitted.

Example 5. *Consider the signing sequence* $(a, b, c,\ a, b, c,\ a, b, c)$*. The sequence is easily seen to be a fair signing sequence. By Theorem 5, the sequence* $(d, a, b, c,\ a, b, c,\ a, b, c,\ d,\ c,\ a, b, c,\ a, b, c)$ *is a fair signing sequence.*

VI. Complexity

There are several common complexity measures for asynchronous, optimistic MPCS protocols, but no common terminology for these measures. One measure is in terms of the total number of messages sent in the optimistic case, when assuming that at each protocol step only one message containing several promises is forwarded to the next signer such as described for the MRT protocols in the previous section. We call this measure *message complexity*.

The other measure concerns the *bandwidth* necessary to execute the protocol in the optimistic case. Since our protocol compiler produces a specification in which every message sent contains exactly one private contract signature or one universally verifiable signature, the bandwidth complexity can be defined in terms of the total number of send instructions produced by our protocol compiler. We will denote the bandwidth of the compiled protocol for σ by $B(\sigma, SigSet)$ and define it to be equal to the number of send instructions appearing in the main protocol specification.

It is known [6] that the minimum message complexity for k signers is $O(k^2)$ and that the minimum bandwidth complexity is $O(k^3)$. The connection between fair MPCS protocols and fair sequences allows us, however, to give more precise bounds.

A further measure commonly used is the *round* complexity [5], [11]. This measure does not have a natural definition for the protocols constructed in this paper, because it assumes the existence of a repeating structure in the protocols.

A. Message complexity

Let $\lambda(k)$ be the length of the shortest complete sequence over a set A with $|A| = k$. It follows from Theorems 1 and 4 that the minimum message complexity of our MPCS protocols is equal to $\lambda(k) + 2k - 3$ and that this bound is *tight*. The term $2k-3$ stems from the $k-1$ signers added to a minimal-length complete sequence for the initial promises in the beginning of the protocol and $k - 2$ signers for the forwarded delivery of the last signer's signature in the end.

It is easy to see that $\lambda(2) = 3$. Newey has shown that for $k > 2$, $\lambda(k) \leq k^2 - 2k + 4$ with equality for $2 < k < 8$. Zălinescu [15] has shown $\lambda(k) \leq k^2-2k+3$ for $k \geq 10$. The presently best known bound to us is $\lambda(k) \leq \lceil k^2 - \frac{7}{3}k + \frac{19}{3} \rceil$ for $k \geq 7$ shown by Radomirović [16]. This bound matches the preceding ones for $7 \leq k < 13$, but is smaller for $k \geq 13$.

The following is an example sequence for a set of 13 elements, $A = 0, \ldots, 9, a, x, y$, matching the currently best known bound. For ease of reading, we do not separate the elements of this sequence by a comma.

$$
\begin{aligned}
x\ 0123456789a\ yx\ 0123456789a\ xy\ 0123456789\ x \\
a01234567y8\ x\ 9a01234567\ x\ 8y9a0123456\ x \\
789a01234y5\ x\ 6789a01234\ x\ 5y6789a0123\ x \\
456789a012\ yx\ 3456789a012\ xy\ 3456789a012\ x
\end{aligned}
$$

B. Bandwidth complexity

While we do not know the precise minimum message complexity of fair MPCS protocols involving $k \geq 8$ signers, we know that it is equal to $\lambda(k)+2k-3$, where $\lambda(k)$ is the subject of a well-known open combinatorial problem. The situation for the precise bandwidth complexity is worse.

A simple upper bound for the minimal bandwidth complexity of fair signing sequences is $(\lambda(k) + 2k - 3)(k - 1)$. This bound is obtained by taking the minimal message complexity of fair MPCS protocols and assuming that in every step of the protocol every signer sends a signature to every other signer. The MR protocols of Section V provide a better bound: $k(k-1)(\lceil k/2 \rceil + 1)$. This is due to the fact that in MR protocols an average of $k/2$ send instructions are specified in a protocol step, whereas the upper bound above assumes $k - 1$ instructions.

By computing the bandwidth for all possible fair signing sequences over 3 and 4 signers, one learns that the signing sequences $(3, 2, 1, 2, 1, 3, 1, 2, 1)$ and $(4, 3, 2, 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1)$ have minimal bandwidth with 14 and 34 signatures, respectively. One recognizes in these sequences the *longest* fair signing sequences with the property that removing any element yields an unfair signing sequence. Such sequences are obtained as follows. The complete part of the sequence is constructed recursively: $\sigma^1 = (1), \sigma^{k+1} = (\sigma^k, k + 1, \sigma^k)$. Completeness is proven as follows. σ^1 is complete over $\{1\}$. If σ^k is complete, then σ^{k+1} is complete, because it contains all permutations over $[k]$ before as well as after the unique appearance of $k + 1$. Thus it contains all permutations over $[k + 1]$. To finally obtain the fair signing sequence, we prepend $(k, \ldots, 2)$ to σ^k. The $SigSet^k$ is $\mathrm{rmpi}(\sigma^k)$.

The bandwidth complexity of these protocols can be bounded below by $B(\sigma^k, SigSet^k) \geq 3 \cdot 2^{k-1} - 2k$. It follows that for sufficiently many signers (say $k > 10$) the bandwidth complexity of this type of sequences is vastly larger than the bandwidth complexity of MR protocols.

To derive the bound, note that the number $s(k)$ of send instructions in the recursively constructed complete part of the signing sequences satisfies $s(2) > 2$ and $s(k + 1) \geq 2s(k) + 2k - 2$. Thus, let $b(x)$ be a function which satisfies $b(2) = 2$ and $b(x + 1) = 2 \cdot b(x) + 2x - 2$. Then it is easy to see that $b(x) = 3 \cdot 2^{x-1} - 2x$.

The construction of minimal bandwidth fair MPCS protocols in general is an open problem.

A general result we can prove is that in order to find a minimal bandwidth fair MPCS protocol, it suffices to consider only fair signing sequences σ with the property that removing any element from σ yields an unfair signing sequence.

Theorem 6. *Let ρ, τ be sequences over A such that their concatenation $\sigma = (\rho, \tau)$ is a signing sequence over A. Then $B(\sigma, SigSet) \leq B((\rho, c, \tau), SigSet')$ for any $c \in A$, where $SigSet'$ is obtained from $SigSet$ by shifting positions and adding the position of c to $SigSet'$, if necessary.*

Proof: It suffices to consider send instructions in the protocol specification for (ρ, τ) which connect a position in ρ to a position in τ and involve c as sender or recipient. Every such instruction will be replaced by an instruction originating or ending at the newly introduced c in the specification for (ρ, c, τ). Thus, there will be at least as many send instructions in the protocol specification for (ρ, c, τ) as in the protocol specification for (ρ, τ). ∎

While the proof above establishes that the bandwidth complexity will be no smaller by introducing a new element into the signing sequence, it is easy to see that it can be strictly larger. Consider, for instance, the sequences (c, a, b, a, c) and (c, a, b, c, a, c). The former sequence has four send instructions involving c while the latter has six.

VII. Generalizations

Our theory extends to the case where fewer dishonest signers can be tolerated in exchange for shorter protocols. We merely give a brief account of this fact.

The controlled reduction in the number of dishonest signers is achieved by requiring that all permutations of t-element subsets of A are present in the sequences, rather than all permutations of A. The protocol compiler then produces sequences which can tolerate up to $t-1$ dishonest signers.

We say that a sequence is *t-complete over A*, if it contains all t-element permutations of elements in A as subsequences. For instance, the sequence $(1, 2, 3, 4, 3, 2, 1)$ is 2-complete over $\{1, 2, 3, 4\}$, since it contains (i, j) as a subsequence for all $i, j \in A$. Therefore, the protocol compiler applied to $\sigma = (4, 3, 2,\ 1, 2, 3, 4, 3, 2, 1)$ with signing set $\mathrm{rmpi}(\sigma)$ produces a fair MPCS protocol, as long as there are fewer than 2 dishonest signers.

Similarly, the sequence $(1, 2, 3, 4, 3, 2, 1, 2, 3, 4)$ is 3-complete over A, thus $\sigma' = (4, 3, 2,\ 1, 2, 3, 4, 3, 2, 1, 2, 3, 4)$ with signing set $\mathrm{rmpi}(\sigma')$ produces a fair MPCS protocol, as long as there are fewer than 3 dishonest signers. A construction for short t-complete sequences has been given by Savage [17].

A closer inspection of our theory reveals that we can give an even more precise characterization of the set of dishonest signers that can be tolerated. Consider the signing sequence $\sigma = (4, 3, 2,\ 1, 2, 3, 4, 3, 2, 1)$ again. The sequence is fair for signers $2, 3, 4$ as long as there is at most one dishonest signer. This is because, using notation of Theorem 1, σ^2, σ^3, and σ^4 are 1-complete over $A \setminus \{2\}$, $A \setminus \{3\}$, and $A \setminus \{4\}$, respectively. But for signer 1 the sequence is fair as long as there are at most two dishonest signers, because σ^1 is 2-complete over $A \setminus \{1\}$.

VIII. Conclusion

We have demonstrated a procedure which allows for a flexible construction of fair optimistic MPCS protocols and we have given several examples and starting points for such constructions. We have proven our construction to be correct and we have investigated the minimum complexity of the generated protocols for two types of complexity measures.

At the heart of our construction lies the generation of sequences which contains all permutations of a finite set as subsequences. This connection has first been noticed in [1] where it has been used to prove a lower bound for the message complexity of fair MPCS protocols. Our results not only confirm the existence of fair protocols with the minimum message complexities established in that prior work, but also show a tighter correspondence between fair optimistic MPCS protocols and sequences which contain all permutations of the signers' set as subsequences. This improvement is due to our generalized notion of fair signing sequences.

Acknowledgments

Barbara Kordy was supported by the National Research Fund, Luxembourg (C08/IS/26).

We thank Jun Pang and the anonymous reviewers for comments that helped us to improve the manuscript.

References

[1] S. Mauw, S. Radomirović, and M. T. Dashti, "Minimal message complexity of asynchronous multi-party contract signing," in *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*. IEEE Computer Society, 2009, pp. 13–25.

[2] S. D. Gordon, "On fairness in secure computation," PhD Thesis, University of Maryland, 2010.

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

[4] 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.

[5] 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.

[6] J. A. Garay and P. D. MacKenzie, "Abuse-free multi-party contract signing," in *Distributed Computing, 13th International Symposium, Bratislava, Slavak Republic, September 27-29, 1999, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1693. Springer, 1999, pp. 151–165.

[7] 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.

[8] Y. Zhang, C. Zhang, J. Pang, and S. Mauw, "Game-based verification of multi-party contract signing protocols," in *Formal Aspects in Security and Trust*, ser. Lecture Notes in Computer Science, P. Degano and J. D. Guttman, Eds., vol. 5983. Springer, 2009, pp. 186–200.

[9] ——, "Game-based verification of multi-party contract signing protocols with minimal messages," *Innovations in Systems and Software Engineering*, in press.

[10] 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, Jul. 2000, pp. 524–535.

[11] 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, Jul. 2001, pp. 898–911.

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

[13] 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.

[14] S. P. Mohanty, "Shortest string containing all permutations," *Discrete Math.*, vol. 31, pp. 91–95, 1980.

[15] E. Zalinescu, "Shorter strings containing all k-element permutations," *Inf. Process. Lett.*, vol. 111, no. 12, pp. 605–608, 2011.

[16] S. Radomirović, "A construction of short sequences containing all permutations of a set as subsequences," unpublished.

[17] C. Savage, "Short strings containing all k-element permutations," *Discrete Math.*, vol. 42, pp. 281–285, 1982.