Formal analysis of multi-party contract signing

Rohit Chadha (rchadha@sussex.ac.uk) *
University of Sussex

Steve Kremer (kremer@lsv.ens-cachan.fr) †
LSV, ENS de Cachan & CNRS, INRIA Futurs Projet SECSI

Andre Scedrov (scedrov@math.upenn.edu) ‡
University of Pennsylvania

Abstract.
We analyze the multi-party contract-signing protocols of Garay and MacKenzie (GM) and of Baum and Waidner (BW). We use a finite-state tool, Mocha, which allows specification of protocol properties in a branching-time temporal logic with game semantics. While our analysis does not reveal any errors in the BW protocol, in the GM protocol we discover serious problems with fairness for four signers and an oversight regarding abuse-freeness for three signers. We propose a complete revision of the GM subprotocols in order to restore fairness.

1. Introduction

The problem of digitally signing a contract over a network is more complicated than signing a contract by "pen and paper". The problem arises because of an inherent asymmetry: no signer wants to be the first one to sign the contract because another signer could refuse to do so after having obtained the first signer's contract.

A simple solution consists in using a trusted party (T) as an intermediary. Signers send their respective contracts to T, which first collects the contracts and then distributes them among the signers. An intermediary is known to be necessary (Even and Yacobi, 1980).

* Partially supported by the ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795 and by the NSF Grant CCR-0098096 and the EU Global Computing project "MIKADO".

† This research was partially carried out while the author stayed at University of Pennsylvania funded by the "Communauté Française de Belgique". Partially supported by the ACI-SI Rossignol, the ACI JC 9005 and the RNTL project PROUVÉ 03V360.

‡ Partially supported by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795 and OSD/ONR CIP/SW URI "Trustworthy Infrastructure, Mechanisms, and Experimentation for Diffuse Computing" through ONR Grant N00014-04-1-0725. Additional support from NSF Grants CCR-0098096 and CNS-0429689.

However, because of the communication and computation bottleneck at T, this solution is inefficient. Other solutions include randomized protocols as well as protocols based on gradual information exchange. More recently, the so-called *optimistic* approach was introduced in (Asokan et al., 1997; Burk and Pfitzmann, 1990). The idea is that T intervenes only when a problem arises, *e.g.,* a signer is trying to cheat or a network failure occurs at a crucial moment during the protocol. Such protocols generally consist of a main protocol and one or several subprotocols, each with a fixed number of messages. The main protocol is executed by the signers in order to exchange their signatures. The subprotocols are used to contact T in order to force a successful outcome or to abort the protocol.

A contract-signing protocol should respect several desirable properties. The first property is *fairness*. Intuitively, a contract-signing protocol is fair if at the end of the protocol either each signer obtains all the other signers' contracts or no signer gets any valuable information. A second property, *timeliness*, ensures that signer has some recourse to prevent endless waiting. Both fairness and timeliness are standard properties that are also important in fair exchange, certified e-mail and fair non-repudiation protocols. A property that is specific to contract signing, *abuse-freeness*, was introduced in (Garay et al., 1999). A protocol is abuse-free if no signer A is able to prove to an external observer that A has the power to choose between successfully concluding the protocol and aborting the protocol. A protocol that is not abuse-free gives an undesirable advantage to one signer, say Alice, who has the power to decide the outcome of the protocol and can prove this to an external observer. If, for instance, Alice wants to sell a house to Charlie, she could initiate a contract with Bob just to force Charlie to increase his offer.

There have been several applications of formal methods to contract signing, so far only for the special case of two signers. The finite model-checker Murφ is used in (Shmatikov and Mitchell, 2002) to analyze two contract-signing protocols, discover subtle errors and suggest corrections. In (Chadha et al., 2001) inductive methods are used to reason about contract-signing protocols specified in the multiset-rewriting framework, MSR. Protocol properties are expressed in terms of strategies, which provide a natural framework for the analysis. In (Kremer and Raskin, 2002) the finite model-checker MOCHA is used to analyze two contract-signing protocols. The advantage of using MOCHA rather than Murφ is that MOCHA allows to specify protocol properties in ATL, a temporal logic with game semantics, which in turn allows reasoning about strategies. In (Gürgens and Rudolph, 2003) the finite state tool SHVT is used to analyze several variants of the Zhou-Gollmann non-

repudiation protocols (non-repudiation protocols are closely related to contract signing protocols). Protocols are modeled using asynchronous product automata and properties are basically invariants. They show unknown attacks which can occur in a realistic implementation of the protocol. The most recent work on contract signing (Chadha et al., 2005) introduces the notion of an *optimistic* signer, *i.e.,* a signer that prefers to wait for "some time" for messages from the other signers before contacting the trusted party. The main theorem of (Chadha et al., 2005) is that, independently of a specific protocol, if any of the signers is optimistic, then the other signer will at some point of the protocol have the power to decide the outcome.

All the efforts just described consider only two-party protocols. In this paper we analyze multi-party contract-signing protocols (Baum-Waidner and Waidner, 2000; Garay and MacKenzie, 1999). The protocol goal in that case is that each signer sends its signature on a previously agreed upon contract text to all other signers and that each signer receives all other signers' signatures on this contract. In a multi-party framework, fairness, timeliness, and abuse-freeness should hold against *any* coalition of dishonest parties. Unlike in the two-party case, the complexity level of the multi-party protocols, especially (Garay and MacKenzie, 1999), is such that a tool, e.g., a model-checker, is indispensable in the analysis. This partly comes about from an important difference between the two-party and the multi-party case, namely, in the multi-party case T has to be able to overturn its previous abort decisions (Garay et al., 1999). As our analysis shows, this feature is particularly difficult to design correctly.

We chose MOCHA for our analysis which allowed specification of properties in ATL, a branching-time temporal logic with game semantics. Apart from our familiarity with MOCHA, two important aspects of optimistic contract signing influenced our choice of the tool. The first is the branching aspect of optimistic contract-signing protocols; the protocols usually consist of subprotocols that can be invoked at specific moments. The other is that the desired properties of contract signing are naturally expressed as winning strategies in games.

For example, one possible formulation of fairness often used in literature (Kremer and Raskin, 2002; Chadha et al., 2005), is that a protocol is fair for an honest signer Alice if whenever some other signer, say Bob, receives Alice's signature on the contract, then Alice has a strategy to get the signed contracts from all other signers. Similarly, a possible formulation for timeliness (Kremer and Raskin, 2002) says that a protocol is timely for Alice if regardless of the state of execution of the protocol, Alice has a strategy to reach a *terminal state* in which she is neither required to send any messages nor expects any messages.

It is in the formulation of abuse-freeness, however, that strategies seem to be indispensable. In (Chadha et al., 2003), for example, a protocol is said to be abuse-free for Alice if it does not provide a provable advantage to the remaining signers. Assuming that Alice always prefers to wait for "some time" before contacting the trusted party, a protocol is said to provide provable advantage to a coalition of signers against Alice at a point in a protocols if a) they have a strategy to abort the exchange, b) they have a strategy to complete the exchange, and c) they can prove to an outside observer that Alice is participating in the protocol. Our formulation of abuse-freeness follows this definition.

Although, ATL does allow elegant formulation of the desired properties in terms of strategies, the failure of an ATL formula cannot be explained by an error-trace in general. Hence the model-checker MOCHA does not provide error traces when these properties fail. Therefore, we looked at other possible formulations of the security properties. For example, we could formulate fairness as a state invariant: a protocol is fair for Alice if in all states terminal for Alice, either Alice has the signed contract or none of the other signers have Alice's signature on the contract. The advantage of this formulation is that MOCHA does provide error traces for invariant checking.

The formulation of fairness as an invariant is strictly weaker than formulation of fairness in terms of strategies mentioned before, and it is relatively easy to construct examples in which fairness in terms of strategies is violated even if the invariant holds. However, in presence of timeliness stated in terms of strategies, these are essentially equivalent. The importance of this result to our work is that since our protocol models satisfied timeliness in terms of strategies, verifying the weaker formulation of fairness in terms of fairness was equivalent to verifying the stronger formulation of fairness in terms of strategies. In (Shmatikov and Mitchell, 2002), Shmatikov and Mitchell use a similar invariant specification of fairness, without however showing the exact relations with other versions of fairness when timeliness is respected.

Our analysis revealed an essential obstacle in the GM protocol (Garay and MacKenzie, 1999), which appears not to be removable without completely changing the subprotocols for T and which leads to the failure of fairness in the case of four signers. In particular, assuming that three signers were dishonest, we found traces leading to a state in which some dishonest signer has the signature of the honest signer, while the honest signer has terminated without having received any signed contracts.

Our analysis also showed that the GM protocol is vulnerable if only two of the signers are dishonest. The dishonest signers may bring the protocol to a state in which an honest signer has terminated without

having received any signed contracts, while the other honest signer has received all the signed contracts. Even though the protocol does not provide an advantage to the dishonest signers in this case, this protocol execution must be viewed as being unfair for the first signer. We did not discover any problems when there was only one dishonest signer.

We present the fairness attacks on the GM protocol in detail in the paper and propose a corrected version of the GM protocol, which has been validated by MOCHA. MOCHA did not find any problems with fairness in the BW protocol (Baum-Waidner and Waidner, 2000) nor in the original GM protocol with only three signers. In the latter case, MOCHA did find an amusing problem with abuse-freeness, but this problem is easily corrected. We believe that the main reason for robustness of the BW protocol is that overturning the abort decisions has been designed correctly.

The rest of the paper is organized as follows. In section 2, we describe the BW and the GM protocols. In section 3, we present briefly the finite-state tool, MOCHA, the temporal logic ATL and its game semantics. Modeling of the protocols and protocol assumptions in the game semantics along with the modeling of fairness and timeliness in ATL is discussed. We also show in this section how the invariant formulation of fairness is related to other formulations of fairness. In section 4, we report on our analysis of the BW and GM protocols using MOCHA, present the fairness attacks on four signers in detail and propose a corrected version of the protocol. We discuss briefly how to restore fairness and present the anomaly with respect to abuse-freeness for three signers. In order to detect this anomaly, we had to model optimistic signers and discuss this issue. We summarize our results and discuss directions for future work in section 5.

2. Protocol description

In this section we describe the multi-party contract-signing protocols proposed by Baum and Waidner in (Baum-Waidner and Waidner, 2000) and Garay and MacKenzie in (Garay and MacKenzie, 1999). Unlike two-party protocols, which generally have similar structures, the two multi-party protocols which we describe below have fundamentally different structures. For this section and for the rest of the paper, we shall assume that each protocol participant has a private signing key and a corresponding public verification key. Each participant shall be identified with this private/public key pair, and if we say that "*A can* ...", we shall mean anyone that possesses the private key of A.

2.1. GM multi-party contract-signing protocol

The protocol allows n ($n \geq 2$) participants, say $P_1, \ldots, P_n$, to exchange signatures with the help of a trusted party T on a preagreed contract text m. P_i is said to have a *contract* if it has everybody's signature on the text m. The order of the participants $P_1, \ldots, P_n$, henceforth referred to as *signers*, and the identity of T are also agreed upon before the protocol begins. The preagreed contract text m contains an identifier that uniquely identifies each protocol instance. In (Garay and MacKenzie, 1999), the communication amongst the participants is assumed to be over a network channel in control of a "Dolev-Yao intruder", while the communication between the participants and the trusted party is assumed to be over a private channel.

The protocol uses zero-knowledge cryptographic primitives, *private contract signatures*, that were first introduced in (Garay et al., 1999). The private contract signature of A for B on text m with respect to a trusted party T, denoted as $PCS_A(m, B, T)$ has the following properties:
a) $PCS_A(m, B, T)$ can be created by A.
b) $PCS_A(m, B, T)$ can be faked by B. Only A, B and T can tell difference between PCS and its simulation.
c) $PCS_A(m, B, T)$ can be converted into a conventional universally-verifiable digital signature, $S_A(m)$, by both A and T. Only A and T can do this conversion.

The protocol itself consists of three subprotocols: *main*, *abort*, and *recovery* subprotocols. Usually signers try to achieve the exchange by executing the main subprotocol. They contact T using one of the other two subprotocols when they think something is amiss. Once a signer contacts T, it no longer takes part in the main subprotocol. T responds to a request with either an abort token or a signed contract. The decision whether to reply with an abort token or with a signed contract is based on a database maintained by T, which stores all the relevant information of the requests and its responses. Once T sends back a signed contract, it always replies with the signed contract. As discussed below, a decision to abort may, however, be overturned in order to maintain fairness. We discuss the subprotocols in some detail.

Main protocol. The main protocol for n signers is divided into n-levels, that can be described recursively. For each level of recursion, a different "strength" of promise is used. The strength of a promise is denoted by an integer "level", and an "i-level promise from signer A to signer B on a message m" is implemented using PCS: $PCS_A((m, i), B, T)$.

In level i, signers P_i through P_1 exchange i-level promises to sign the contract. The i-level protocol is triggered when P_i receives 1-level promises from $P_{i+1}, \ldots, P_n$. After receiving these promises, P_i sends out its 1 level promise to signers $P_{i-1}, \ldots, P_1$ and waits for $i-1$ level to finish. At the end of the $i-1$ level, P_1, $\ldots$, P_{i-1} have exchanged $i-1$ level promises and P_i receives a $i-1$ level promise from each of the signers $P_1, \ldots, P_{i-1}$. Now $P_i \ldots, P_1$ exchange i level promises, and close the higher levels.

In order to close level a where $a > i$, P_i sends an $(a-1)$-level promise to P_a and waits for a-level promises from signers $P_{i+1}, \ldots, P_a$. After receiving these promises, P_i indicates its willingness to close the level a to signers $P_1, \ldots, P_{i-1}$ by sending them its a-level promise, and in return waits for a-level promises from them. Upon receiving these, P_i sends its a-level promises to $P_{i+1}, \ldots, P_a$ completing its obligation in the a-level protocol. P_i then proceeds to complete $a+1$ level.

Once the n-levels are completed, each signer has a n-level promise from everybody else, and the contract exchange is ready to begin. In this exchange, each signer also sends a $n+1$-level promise to everybody along with its signature on the preagreed text. In order to complete the exchange, signer P_i waits for the contract and $n+1$-level promises from $P_n, \ldots, P_{i+1}$. Upon receiving these, P_i sends its signature and $n+1$-level promises to everybody, and waits for the signatures and $n+1$-level promises from $P_{i-1}, \ldots, P_1$. Once these are received, the protocol ends for P_i, and P_i has the contract.

If some expected messages are not received, P_i may either quit the protocol or contact T. P_i may simply quit the protocol if it has not sent any promises or contact T if it has sent some promises. It may contact T with a request to abort if it has not received any promise from some signer. It may request T to recover the protocol if it has a promise from every other signer. A detailed description of the main protocol is given in table 1.

In order to illustrate the main protocol, consider an instance of the protocol with three signers: Alice, Bob and Carol playing the roles of P_3, P_2 and P_1 respectively. For lack of space, we just illustrate the role of Alice. Alice starts the protocol by sending level 1 promises to Bob and Carol, and waits for level 2 promises from Bob and Carol. If Alice does not receive them, then Alice may contact T with a request to abort the exchange. If Alice does receive the promises, then she sends her level 3 promises to Bob and Carol, and waits for their level 3 promises in return. If Alice does not receive these promises then she contact T with a recovery request. Otherwise, she sends her signature on the preagreed text along with level 4 promises to Bob and Carol, and waits for their

signatures. The protocol finishes for her when she receives the contract. Otherwise, she may launch the recovery subprotocol and contact T.

Table 1 GM multi-party contract-signing protocol—Main for P_i

Wait for all higher recursive levels to start

1. $P_j \rightarrow P_i$: $PCS_{P_j}((m,1),P_i,T)$ $(n \geq j > i)$

   If P_i does not receive 1-level promises from $P_n \ldots P_{i+1}$ in a timely manner, then P_i simply quits.

Start recursive level i

2. $P_i \rightarrow P_j$: $PCS_{P_i}((m,1),P_j,T)$ $(i > j \geq 1)$

Wait for recursive level i-1 to finish

3. $P_j \rightarrow P_i$: $PCS_{P_j}((m,i-1),P_i,T)$ $(i > j \geq 1)$

   If P_i does not receive (i-1)-level promises from $P_{i-1} \ldots P_1$ in a timely manner, then P_i aborts.

Send i-level promises to all lower-numbered signers

4. $P_i \rightarrow P_j$: $PCS_{P_i}((m,i),P_j,T)$ $(i > j \geq 1)$

Finish recursive level i when i-level promises are received

5. $P_j \rightarrow P_i$: $PCS_{P_j}((m,i),P_i,T)$ $(i > j \geq 1)$

   If P_i does not receive i-level promises from $P_{i-1} \ldots P_1$ in a timely manner, then P_i recovers.

Complete all higher recursive levels

For $a = i+1$ to n, P_i does the following:

6.1. $P_i \rightarrow P_a$: $PCS_{P_i}((m,a-1),P_a,T)$
6.2. $P_j \rightarrow P_i$: $PCS_{P_j}((m,a),P_i,T)$ $(a \geq j > i)$
If P_i does not receive a-level promises from $P_a \ldots P_{i+1}$ in a timely manner, then P_i recovers.
6.3. $P_i \rightarrow P_j$: $PCS_{P_i}((m,a),P_j,T)$ $(i > j \geq 1)$
6.4. $P_j \rightarrow P_i$: $PCS_{P_j}((m,a),P_i,T)$ $(i > j \geq 1)$
If P_i does not receive a-level promises from $P_{i-1} \ldots P_1$ in a timely manner, then P_i recovers.
6.5. $P_i \rightarrow P_j$: $PCS_{P_i}((m,a),P_j,T)$ $(a \geq j > i)$

Wait for signatures and (n+1)-level promises from higher-numbered signers

7. $P_j \rightarrow P_i$: $PCS_{P_j}((m,n+1),P_i,T), S_{P_j}(m,1)$ $(n \geq j > i)$

   If P_i does not receive signatures and (n+1)-level promises from $P_n \ldots P_{i+1}$ in a timely manner, then P_i recovers.

Send signatures and (n+1)-level promises to signers

8. $P_i \rightarrow P_j$: $PCS_{P_i}((m,n+1),P_j,T), S_{P_i}(m,1)$ $(j \neq i)$

Wait for signatures from lower-numbered signers

9. $P_j \rightarrow P_i$: $PCS_{P_j}((m,n+1),P_i,T), S_{P_j}(m,1)$ $(i > j \geq 1)$

   If P_i does not receive signatures and (n+1)-level promises from $P_{i-1} \ldots P_1$ in a timely manner, then P_i recovers.

Abort protocol. T maintains two sets, $S(m)$ and $F(m)$, that are used by T to make decisions when a signer contacts T. These sets are created when T is contacted for the first time for m and are initialized to be empty. The set $S(m)$ contains the indices of all signers that have contacted T and received an abort token from T in response. The intuitive meaning of the set $F(m)$ is not clearly stated in (Garay and MacKenzie, 1999), but it contains some additional information that T uses in deciding when to overturn an abort decision that T has taken before.

The details of the abort protocol are given in table 2. Mainly, if T is contacted with a request to abort, then T checks its database. If this is the first request or if the protocol has not already been recovered, T sends back an abort token and updates the sets $S(m)$ and $F(m)$. If the protocol has already been successfully recovered, T sends back a signed contract.

Table 2 GM multi-party contract-signing protocol—Abort for P_i

The first time T is contacted for contract m (either abort or recovery), T initializes $S(m)$ and $F(m)$ to $\emptyset$ and validated(m) to false.

1. $P_i \rightarrow$ T: $S_{P_i}(m, P_i, (P_1, \ldots, P_n), abort)$

if not validated(m) then

- if $S(m) = \emptyset$ then
  - T stores $S_T(S_{P_i}(m, P_i, (P_1, \ldots, P_n), abort))$;
  - $S(m) = S(m) \cup \{i\}$;
- if i is larger than the maximum index in $S(m)$ then
  - T clears $F(m)$
- 2. $T \rightarrow P_i$: $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort), S_T(m, S(m), abort))$ where $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort))$ corresponds to the stored abort token

else (validated(m)=true)

- 3. $T \rightarrow P_i$: $\{S_{P_j}((m, k_j))\}_{j \in \{1,\ldots,n\} \setminus \{i\}}$ where k_j is the level of the promise from P_j that was converted to a universally-verifiable signature during the recovery protocol.

Recovery protocol. The details of the recovery protocol are given in table 3. For P_i to recover, it sends the message

$$S_{P_i}(\{PCS_{P_j}((m, k_j), P_i, T)\}_{j \in \{1,\ldots,n\} \setminus \{i\}}, S_{P_i}((m, 1)))$$

where

- if $j > i$, k_j is the maximum level of a promise received from P_j on m,

– if $j < i$, k_j is the maximum level of promises received from all signers $P_{j'}$, with $j' < i$, *i.e.*, the min-max of the level of promises from signers with lower index. (*E.g.*, if the maximum level of the promises received by P_4 from P_3 and P_2 was 6, and the maximum level received by P_4 from P_1 was 5, then it would send the 5-level promises for P_1, P_2 and P_3.)

If T is contacted with a request to recover, then T checks its database. If this is the first request for m or if the protocol has already been recovered, T replies with a signed contract which it obtains by converting the promises into conventional digital signatures. Otherwise, if the protocol has already been aborted, T must decide whether to maintain the abort or to overturn it. Overturning of the abort is necessary in order to maintain fairness. Indeed, consider the scenario in which a dishonest P_{n-1} contacts T with an abort request, receives an abort token and dishonestly continues the protocol. After the n-levels are completed, P_n sends its signature to others and waits for signatures from other signers. If P_{n-1} does not send back its signature, then P_n will be forced to contact T with a request to recover. Now, T must overturn its previous abort, otherwise P_n will not receive the signature of P_{n-1}. The decision whether to overturn is based on the contents of the sets $S(m)$ and $F(m)$, as described in table 3.

2.2. BW multi-party contract-signing protocol

The protocol allows n ($n \geq 2$) participants or *signers*, say $P_1, \ldots, P_n$, to exchange signatures with the help of a trusted party T on a preagreed contract text m. In our description, we suppose $n-1$ potentially dishonest signers. The original protocol is actually parameterized with respect to a threshold t, the maximum number of possibly dishonest signers. In our analysis however we assume the worst possible scenario for an honest signer, namely that all the other signers are dishonest (*i.e.*, t is $n-1$). We assume the same hypotheses on the network as for the GM protocol.

The protocol consists of two subprotocols: main and recovery. Usually signers try to achieve the exchange by executing the main subprotocol. They contact T using the recovery subprotocol when they think something is amiss.

Main protocol. The main protocol for each signer P_i is given in table 4. The protocol is symmetric for each signer and is composed of $n+1$ rounds[1]. In each round, each signer sends a promise to other signers.

[1] In (Baum-Waidner and Waidner, 2000), the protocol has $t+2$ rounds.

Table 3 GM multi-party contract-signing protocol—Recovery for P_i

The first time T is contacted for contract m (either abort or recovery), T initializes $S(m)$ and $F(m)$ to $\emptyset$ and validated(m) to false.

1. $P_i \rightarrow$ T: $S_{P_i}(\{PCS_{P_j}((m,k_j),P_i,T)\}_{j\in\{1,\ldots,n\}\setminus\{i\}}, S_{P_i}((m,1)))$

if $i \in S(m)$ then

  T ignores the message

else if validated(m) then

  2. $T \rightarrow P_i$: $\{S_{P_j}((m,k_j))\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$
  where k_j is the level of the promise from P_j that was converted to a universally-verifiable signature.

else if $S(m) = \emptyset$ then

  validated(m):=true
  3. $T \rightarrow P_i$: $\{S_{P_j}((m,k_j))\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$

else (validated(m)=false $\wedge$ $S(m) \neq \emptyset$)

  if $i \notin F(m)$ then
    if for any $\ell \in S(m)$ there is a $j \in S(m)$ such that $j > k_\ell$ then
      $S(m) := S(m) \cup \{i\}$
      let a be the maximum value in $S(m)$
      if $a > i$ then $\forall j$, such that $k_j = a - 1 \cdot$ $F(m) := F(m) \cup \{j\}$
      else $F(m) := \emptyset$
      4.1.1. $T \rightarrow P_i$: $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort), S_T(m, S(m), abort))$
      where $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort))$ corresponds to the stored abort token
    else
      validated(m):=true
      4.1.2. $T \rightarrow P_i$: $\{S_{P_j}((m,k_j))\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$
  else ($i \in F(m)$)
    let a be the maximum value in $S(m)$
    if ($\forall j$, such that $i < j \leq a \cdot$ $k_j < a$) $\wedge$ ($\forall j < i \cdot$ $k_j \geq a$) then
      validated(m):=true
      4.2.1. $T \rightarrow P_i$: $\{S_{P_j}((m,k_j))\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$
    else
      $S(m) := S(m) \cup \{i\}$
      if $a > i$ then $\forall j, k_j = a - 1 \cdot$ $F(m) := F(m) \cup \{j\}$
      if $a = i$ then $F(m) := \emptyset$
      4.2.2. $T \rightarrow P_i$: $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort), S_T(m, S(m), abort))$
      where $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort))$ corresponds to the stored abort token

Table 4 BW multi-party contract signing protocol—Main for P_i

$r := 1$

1. $P_i \rightarrow P_j$: $m_{1,i} = S_{P_i}(m, 1, \mathsf{prev\_round\_ok})(j \neq i)$
2. $P_j \rightarrow P_i$: $m_{1,j} = S_{P_j}(m, 1, \mathsf{prev\_round\_ok})$ $(j \neq i)$

if P_i times out then
  recovery(1)
else
  P_i computes vectors $M_{1,i} := (m_{1,1}, \ldots, m_{1,n})$ and $X_{1,i} := M_{1,i}$
  for $r := 2$ to $n + 1$ do
    3. $P_i \rightarrow P_j$: $m_{r,i} = S_{P_i}(M_{r-1,i}, r, \mathsf{vec\_ok}), S_{P_i}(m, r, \mathsf{prev\_round\_ok})(i \neq j)$
    4. $P_j \rightarrow P_i$: $m_{r,j} = S_{P_j}(M_{r-1,j}, r, \mathsf{vec\_ok}), S_{P_j}(m, r, \mathsf{prev\_round\_ok})$ $(j \neq i)$
    if P_i times out then
      recovery(r)
    else
      P_i computes vectors
        $M_{r,i} := (S_{P_1}(m, r, \mathsf{prev\_round\_ok}), \ldots, S_{P_n}(m, r, \mathsf{prev\_round\_ok}))$
      and
        $X_{r,i} := (S_{P_1}(M_{r-1,i}, r, \mathsf{vec\_ok}), \ldots, S_{P_n}(M_{r-1,i}, r, \mathsf{vec\_ok}))$

The level of the promise is increased in each round, and considered as a signed contract once the round number equals $n + 1$. The promise is implemented using a universally-verifiable digital signature which includes the history of all previously received promises, through the vectors M and X, as defined in table 4. If any expected message is not received, P_i can decide to launch a recovery protocol.

Recovery Protocol The details of the recovery protocol are given in table 5. If the recovery request is launched in the first round, *i.e.*, P_i did not receive a message from all the signers, the recovery request consists of the first level promise of P_i. Otherwise, if $r > 1$, the recovery request contains, via the vector $X_{r-1,i}$ (see the main subprotocol in table 4), the set of received messages until round $r - 1$, including the $r - 1$ promises from all the other signers.

T maintains a variable, recovered(m), that indicates whether the given contract has been successfully recovered or not. It also maintains a set con(m), containing the indices of the signers that contacted T for m, and a set abort_set(m) containing the indices of the signers for whom T aborted the protocol.

T ignores a recovery request from a signer if the signer has contacted T in the past. Otherwise, it checks whether the contract has already been successfully recovered or not. A successful recovery is always main-

Table 5 BW multi-party contract signing protocol—Recovery for P_i

The first time T is contacted for contract m, T initializes con(m) and abort_set(m) to $\emptyset$ and recovered(m) to false.

1. $P_i \rightarrow$ T: $\mathsf{resolve}_{\mathsf{r,i}}$

where

$$\mathsf{resolve}_{\mathsf{r,i}} = \begin{cases} (1, i, S_{P_i}(m_{1,i}, \mathsf{resolve})) & \text{if } r = 1 \\ (r, i, S_{P_i}(X_{r-1,i}, \mathsf{resolve}), X_{r-1,i}, M_{r-2,i}) & \text{otherwise} \end{cases}$$

and

$$M_{0,i} = \mathsf{nil}$$

if $i \in$ con(m) then

  T ignores the message

else if recovered(m) then

  con(m):=con(m)$\cup\{i\}$
  2. $T \rightarrow P_i$: $\mathsf{signed}_{\mathsf{r,i}}$
  where $\mathsf{signed}_{\mathsf{r,i}}$ = first_signed

else ($\neg$recovered(m))

  if $r = 1$ then
    abort_set(m):=abort_set(m) $\cup\{(r, i)\}$
    con(m):=con(m)$\cup\{i\}$
    3. $T \rightarrow P_i$: $\mathsf{aborted}_{\mathsf{r,i}}$
    where $\mathsf{aborted}_{\mathsf{r,i}} = S_T(m, r, i, \mathsf{aborted})$
  else ($r > 1$)
    if $\forall(s, k) \in$ abort_set(m) $\cdot\, s < r - 1$ then
      if con(m)=$\emptyset$ then first_signed:=($\mathsf{resolve}_{\mathsf{r,i}}, S_T(m, r, i, \mathsf{recovered})$)
      recovered(m):=true
      con(m):=con(m)$\cup\{i\}$
      4. $T \rightarrow P_i$: $\mathsf{signed}_{\mathsf{r,i}}$
      where $\mathsf{signed}_{\mathsf{r,i}}$ = first_signed
    else
      abort_set(m):=abort_set(m) $\cup\{(r, i)\}$
      con(m):=con(m)$\cup\{i\}$
      5. $T \rightarrow P_i$: $\mathsf{aborted}_{\mathsf{r,i}}$
      where $\mathsf{aborted}_{\mathsf{r,i}} = S_T(m, r, i, \mathsf{aborted})$

tained. Otherwise, there are two cases. If the recovery request is sent in the first round, T must abort the protocol, as the request does not contain a proof that all signers actually started the protocol. If the recovery request is sent during any later round, say r, then T checks if all the requests that were aborted previously occurred at least two

rounds before. If so, T can be sure that all the issued abort tokens were sent to signers who dishonestly continued the main protocol. This is because the recovery request contains $r-1$ promises from all these signers, which they are not allowed to send if they contacted T in round $r-2$ or before. Hence, the previous abort decision is overturned. Otherwise, T replies with an abort token.

3. Model

In this section, we discuss modeling of the protocols, and specification of the desired security properties. We begin by describing a game-variant of Kripke structures used to model the protocols, and a branching time temporal logic used to model the desired security guarantees.

3.1. ATS, ATL and Mocha

The desired properties of contract signing are easily described using games, and hence we chose a game-variant of Kripke structures, *alternating transition systems* (ATS) (Alur et al., 1997), to model the protocols. *Alternating-time temporal logic* (ATL) (Alur et al., 1997), is used to reason about alternating transition systems. We use ATL to express the protocol properties of the contract-signing protocols.

Alternating transition systems. Intuitively, an ATS is a Kripke structure with additional information about players.

Definition 1. An *alternating transition system* is a 6-tuple

$$S = \langle \Pi, \Sigma, Q, Q_0, \pi, \delta \rangle$$

with the following components:

- Π is a finite set of propositions.
- Σ is a finite set of players.
- Q is a finite set of states.
- $Q_0 \subseteq Q$ is a set of initial states.
- The *labeling function* $\pi : Q \rightarrow 2^{\Pi}$ maps a state to a set of propositions.

– The *game transition function* $\delta : Q \times \Sigma \to 2^{2^Q} \setminus \{\emptyset\}$ maps a state and a player to a nonempty set of choices, where each choice is a set of possible next states. Furthermore, if $\Sigma = \{a_1, \ldots, a_n\}$, then for every state $q \in Q$ and each possible $Q_1, \ldots, Q_n$ where $Q_i \in \delta(q, a_i)$, $Q_1 \cap \ldots \cap Q_n$ is a singleton.

Definition 2. For two states q and q' and a player a, we say that q' is an *a-successor* of q if there exists a set $Q' \in \delta(q, a)$ such that $q' \in Q'$.

We denote by $succ(q, a)$ the set of a-successors of q.

Definition 3. For two states q and q', we say that q' is a *successor* of q if for all players $a \in \Sigma$, we have $q' \in succ(q, a)$.

Thus, q' is a successor of q if and only if whenever the system S is in state q, the players in Σ can cooperate so that q' will be the next state.

Definition 4. A *computation* of S is an infinite sequence $\lambda = q_0 q_1 q_2 \ldots$ of states such that for all positions $i \geq 0$, the state q_{i+1} is a successor of the state q_i.

We refer to a computation starting at state q as a *q-computation*. For a computation λ and a position $i \geq 0$, we use $\lambda[i]$, $\lambda[0, i]$, and $\lambda[i, \infty]$ to denote the i-th state in λ, the finite prefix $q_0 q_1 \ldots q_i$ of λ, and the infinite suffix $q_i q_{i+1} \ldots$ of λ, respectively.

Alternating-time temporal logic. The *Alternating-time Temporal Logic* (Alur et al., 1997) (ATL for short) is defined with respect to a finite set Π of *propositions* and a finite set Σ of *players*.

Definition 5. Given a finite set Π of *propositions*, and a finite set Σ of *players*, an ATL *formula* is one of the following:

– p, for propositions $p \in \Pi$.

– $\neg\varphi$ or $\varphi_1 \vee \varphi_2$, where φ, φ_1, and φ_2 are ATL formulae.

– $\langle\langle A \rangle\rangle \bigcirc \varphi$, $\langle\langle A \rangle\rangle \Box \varphi$, or $\langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$, where $A \subseteq \Sigma$ is a set of players, and φ, φ_1, and φ_2 are ATL formulae.

The operator $\langle\langle\ \rangle\rangle$ is a *path quantifier*, and $\bigcirc$ ("next"), $\Box$ ("always"), and $\mathcal{U}$ ("until") are *temporal operators*. ATL formulae are interpreted over the states of a given ATS, say S, that has the same propositions and players. The labeling of the states of S with propositions is used to evaluate the atomic formulae of ATL. The logical connectives $\neg$ and $\vee$

have the standard interpretation. To give an idea of how to evaluate a formula of the form $\langle\langle A\rangle\rangle\psi$ at a state q of S, consider the following two-player game between a protagonist and an antagonist starting in q. The game proceeds in an infinite sequence of steps, and after each step, the position of the game is a state of S. Consider the game in some position u. In order to determine the next state, first the protagonist chooses for every player $a \in A$, a set $Q_a \in \delta(u, a)$. Then, the antagonist chooses a successor (with respect to definition 3) v of u such that $v \in \cap_{a\in A} Q_a$, and the position of the game is updated to v. The protagonist wins the game if the resulting computation satisfies the subformula ψ, read as a linear temporal formula whose outermost operator is $\bigcirc$, $\Box$, or $\mathcal{U}$. The ATL formula $\langle\langle A\rangle\rangle\psi$ holds at the state q if the protagonist has a *winning strategy* in this game. For those familiar with branching time temporal logics, the parameterized path quantifier $\langle\langle A\rangle\rangle$ can be seen as a generalization of the path quantifiers of the computation tree logic (CTL): the existential path quantifier $\exists$ corresponds to $\langle\langle\Sigma\rangle\rangle$ and the universal path quantifier $\forall$ corresponds to $\langle\langle\emptyset\rangle\rangle$.

In order to give a formal definition of the semantics of ATL, we first define the notion of strategies. Please note that in the definition, given a set Q of states, Q^* is the set of all finite (possibly empty) sequence of states in Q. Q^+ is the set of all non-empty finite sequence of states.

Definition 6. Consider an ATS $S = \langle\Pi, \Sigma, Q, Q_0, \pi, \delta\rangle$. A *strategy* for a player $a \in \Sigma$ is a mapping $f_a : Q^+ \to 2^Q$ such that for all $\lambda \in Q^*$ and all $q \in Q$, we have $f_a(\lambda \cdot q) \in \delta(q, a)$.

For a set of players $A \subseteq \Sigma$, a *strategy function* $F_A = \{f_a | a \in A\}$ is a set of strategies, one for each player in A.

Thus, the strategy f_a maps each finite prefix $\lambda \cdot q$ of a computation to a set in $\delta(q, a)$ and induces a set of computations that player a can enforce. The strategy function F_A induces a set of computations that the set of players A can collaboratively enforce. Given a state q, and a strategy function F_A, we define the *outcomes* of F_A from q to be the set $out(q, F_A)$ of all q-computations that the players in A can enforce by following their strategies in F_A. More precisely, we say that a computation $\lambda = q_0, q_1, q_2, \ldots$ is in $out(q, F_A)$ if $q_0 = q$, and for all positions $i \geq 0$, the state q_{i+1} is a successor of q_i satisfying $q_{i+1} \in \bigcap_{f_a\in F_A} f_a(\lambda[0, i])$.

We are now ready to define the semantics of ATL. We write $S, q \models \varphi$ ("state q satisfies formula φ in the structure S") to indicate that the formula φ holds at state q of S.

Definition 7. Given an ATS, $S = \langle\Pi, \Sigma, Q, Q_0, \pi, \delta\rangle$, and ATL formulae φ, φ_1 and φ_2 over propositions in Π and players in Σ. The relation $\models$ for all $q \in Q$ is defined inductively as follows:

- For $p \in \Pi$, we have $S, q \models p$ iff $p \in \pi(q)$.
- $S, q \models \neg\varphi$ iff $S, q \not\models \varphi$.
- $S, q \models \varphi_1 \vee \varphi_2$ iff $S, q \models \varphi_1$ or $S, q \models \varphi_2$.
- $S, q \models \langle\langle A\rangle\rangle \bigcirc \varphi$ iff there exists a strategy function F_A, such that for all computations $\lambda \in out(q, F_A)$, we have $S, \lambda[1] \models \varphi$.
- $S, q \models \langle\langle A\rangle\rangle \Box\varphi$ iff there exists a strategy function F_A, such that for all computations $\lambda \in out(q, F_A)$ and all positions $i \geq 0$, we have $S, \lambda[i] \models \varphi$.
- $S, q \models \langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a strategy function F_A, such that for all computations $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ (depending on λ) such that $S, \lambda[i] \models \varphi_2$ and for all positions $0 \leq j < i$, we have $S, \lambda[j] \models \varphi_1$.

When S is clear from the context we omit it and write $q \models \varphi$. Sometimes we write $\langle\langle a_1, \ldots, a_n\rangle\rangle$ instead of $\langle\langle\{a_1, \ldots, a_n\}\rangle\rangle$. Additional boolean connectives, $\wedge, \rightarrow, \leftrightarrow$ are defined from $\neg$ and $\vee$ in the usual manner. As in CTL, we write $\langle\langle A\rangle\rangle \Diamond\varphi$ for $\langle\langle A\rangle\rangle true \mathcal{U} \varphi$.

Please note that in the special case of $\forall\Box\varphi \equiv \langle\langle\emptyset\rangle\rangle\Box\varphi$, the definition of $\models$ tells us that $S, q \models \forall\Box\varphi$ if and only if for all q-computations λ, and for all positions $i \geq 0$, we have $S, \lambda[i] \models \varphi$. Similarly, $S, q \models \exists\Diamond\varphi$ if and only if there is some q-computation λ and some position $i \geq 0$, such that $S, \lambda[i] \models \varphi$. As in CTL, we have the duality between $\forall\Box$ and $\exists\Diamond$: $S, q \models \neg\forall\Box\varphi$ if and only if $S, q \models \exists\Diamond\neg\varphi$. Note however that in ATL $S, q \models \langle\langle A\rangle\rangle\Box\varphi \Rightarrow S, q \models \neg\langle\langle\Sigma \setminus A\rangle\rangle\Diamond\neg\varphi$ and $S, q \models \langle\langle A\rangle\rangle\Diamond\varphi \Rightarrow S, q \models \neg\langle\langle\Sigma \setminus A\rangle\rangle\Box\neg\varphi$ while the converse statements are not true in general.

We now illustrate the expressive power of ATL. Consider the set of players $\Sigma = \{a, b, c\}$ and the following formulae with their verbal (and intuitive) reading:

- $\langle\langle a\rangle\rangle\Diamond p$, player a has a strategy against players b and c to eventually reach a state where the proposition p is true;
- $\neg\langle\langle b, c\rangle\rangle\Box p$, the coalition of players b and c does not have a strategy against a such that the proposition p remains true forever;
- $\langle\langle a, b\rangle\rangle\bigcirc(p \wedge \neg\langle\langle c\rangle\rangle\Box p)$, a and b can cooperate so that the next state satisfies p and from there c does not have a strategy to impose p forever.

Those three formulae are a good illustration of the great expressive power of ATL to express cooperative as well as adversarial behaviors between players.

We now introduce a proposition stating several facts about ATL. These facts are merely of technical interest and are introduced because they will be used in some proofs in section 3.3. While stating the proposition, we shall use an abbreviation familiar from logic. In order to say that P is true whenever $P_1, \ldots P_n$ is true, we shall write:

$$\frac{P_1 \; P_2 \; \ldots \; P_n}{P}$$

Proposition 1. For any ATS S and any state q of S we have the following.

$$\frac{S, q \models \forall\Box\varphi_1 \wedge \forall\Box\varphi_2}{S, q \models \forall\Box(\varphi_1 \wedge \varphi_2)} \tag{1}$$

$$\frac{S, q \models \langle\langle A\rangle\rangle\Box\varphi_1 \vee \langle\langle A\rangle\rangle\Box\varphi_2}{S, q \models \langle\langle A\rangle\rangle\Box(\varphi_1 \vee \varphi_2)} \tag{2}$$

$$\frac{S, q \models \forall\Box((\phi \wedge \varphi_1) \rightarrow \forall\Box\psi_1) \quad S, q \models \forall\Box((\phi \wedge \varphi_2) \rightarrow \forall\Box\psi_2)}{S, q \models \forall\Box((\phi \wedge (\varphi_1 \vee \varphi_2) \rightarrow \forall\Box(\psi_1 \vee \psi_2)} \tag{3}$$

$$\frac{S, q \models \forall\Box(\varphi_1 \rightarrow \varphi_2) \quad S, q \models \langle\langle A\rangle\rangle\Diamond(\varphi_1 \wedge \varphi_3)}{S, q \models \langle\langle A\rangle\rangle\Diamond(\varphi_2 \wedge \varphi_3)} \tag{4}$$

$$\frac{S, q \models \forall\Box(\varphi_1 \rightarrow \forall\Box\psi_1) \quad S, q \models \forall\Box(\varphi_2 \rightarrow \forall\Box\psi_2)}{S, q \models \forall\Box((\varphi_1 \wedge \varphi_2) \rightarrow \forall\Box(\psi_1 \wedge \psi_2))} \tag{5}$$

$$\frac{S, q \models \forall\Box(\varphi_1 \rightarrow \forall\Box\varphi_1) \quad S, q \models \forall\Box(\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)) \quad S, q \models \forall\Box(\langle\langle A\rangle\rangle\Diamond\varphi_2)}{S, q \models \forall\Box(\varphi_1 \rightarrow \langle\langle A\rangle\rangle\Diamond\varphi_3)} \tag{6}$$

Proof. We shall use the definition of $\models$ to prove the above.

1. We have that $S, q \models \forall\Box\varphi_1 \wedge \forall\Box\varphi_2$, *i.e.*, $S, q \models \forall\Box\varphi_1$ and $S, q \models \forall\Box\varphi_2$. Let λ be an arbitrary q-computation of S, and $i \geq 0$ be an arbitrary position. Since $S, q \models \forall\Box\varphi_1$, we have $S, \lambda[i] \models \varphi_1$. Similarly, we also have $S, \lambda[i] \models \varphi_2$. Hence, we get $S, \lambda[i] \models \varphi_1 \wedge \varphi_2$. Since λ is an arbitrary q-computation, and i an arbitrary position, we get $S, q \models \forall\Box(\varphi_1 \wedge \varphi_2)$.

2. We have that $S, q \models \langle\langle A\rangle\rangle\Box\varphi_1 \vee \langle\langle A\rangle\rangle\Box\varphi_2$, *i.e.*, $S, q \models \langle\langle A\rangle\rangle\Box\varphi_1$ or $S, q \models \langle\langle A\rangle\rangle\Box\varphi_2$. Suppose $S, q \models \langle\langle A\rangle\rangle\Box\varphi_1$. Then there exists a strategy function F_A, such that for all computation $\lambda \in out(q, F_A)$, and for every position $i \geq 0$, $S, \lambda[i] \models \varphi_1$, and hence $S, \lambda[i] \models \varphi_1 \vee \varphi_2$. Therefore, we obtain $S, q \models \langle\langle A\rangle\rangle\Box(\varphi_1 \vee \varphi_2)$ in this case. Similarly, if $S, q \models \langle\langle A\rangle\rangle\Box\varphi_2$, we shall obtain $S, q \models \langle\langle A\rangle\rangle\Box(\varphi_1 \vee \varphi_2)$.

3. We have that

$$\begin{array}{ll} & S, q \models \forall\Box((\phi \wedge \varphi_1) \rightarrow \forall\Box\psi_1) \wedge \forall\Box((\phi \wedge \varphi_2) \rightarrow \forall\Box\psi_2) \\ \Rightarrow & S, q \models \forall\Box(((\phi \wedge \varphi_1) \rightarrow \forall\Box\psi_1) \wedge ((\phi \wedge \varphi_2) \rightarrow \forall\Box\psi_2)) \\ & \text{by rule (1)} \\ \Rightarrow & S, q \models \forall\Box(((\phi \wedge \varphi_1) \vee (\phi \wedge \varphi_2)) \rightarrow (\forall\Box\psi_1 \vee \forall\Box\psi_2)) \\ & \text{by the boolean law } (p \rightarrow q) \wedge (r \rightarrow s) \Rightarrow (p \vee r) \rightarrow (q \vee s) \\ \equiv & S, q \models \forall\Box((\phi \wedge (\varphi_1 \vee \varphi_2)) \rightarrow (\forall\Box\psi_1 \vee \forall\Box\psi_2)) \\ & \text{by the boolean law} (p \wedge q) \vee (p \wedge r) \equiv p \wedge (q \vee r) \\ \Rightarrow & S, q \models \forall\Box((\phi \wedge (\varphi_1 \vee \varphi_2)) \rightarrow \forall\Box(\psi_1 \vee \psi_2)) \\ & \text{by rule (2)} \end{array}$$

4. We have that $S, q \models \forall\Box(\varphi_1 \rightarrow \varphi_2)$. Hence for every q-computation λ of S, for every position $i \geq 0$, $S, \lambda[i] \models \varphi_1 \rightarrow \varphi_2$.

   Moreover, we also have $S, q \models \langle\langle A\rangle\rangle\Diamond(\varphi_1 \wedge \varphi_3)$. Therefore, there is a strategy function F_A, such that for all computations $\lambda_A \in out(q, F_A)$, there exists a position $j \geq 0$ (depending on λ_A) with $S, \lambda_A[j] \models \varphi_1 \wedge \varphi_3$.

   Now pick $\lambda' \in out(q, F_A)$, and fix it. There is some $k \geq 0$ such that $S, \lambda'[k] \models \varphi_1 \wedge \varphi_3$. That is $S, \lambda'[k] \models \varphi_1$ and $S, \lambda'[k] \models \varphi_3$. We also have that $S, \lambda'[k] \models \varphi_1 \rightarrow \varphi_2$. Hence, we get that $S, \lambda'[k] \models \varphi_2 \wedge \varphi_3$. Since λ' is an arbitrary computation in $out(q, F_A)$, we can conclude that $S, q \models \langle\langle A\rangle\rangle\Diamond(\varphi_2 \wedge \varphi_3)$.

5. We have that

$$\begin{array}{ll} & S, q \models \forall\Box(\varphi_1 \rightarrow \forall\Box\psi_1) \wedge \forall\Box(\varphi_2 \rightarrow \forall\Box\psi_2) \\ \Rightarrow & S, q \models \forall\Box((\varphi_1 \rightarrow \forall\Box\psi_1) \wedge (\varphi_2 \rightarrow \forall\Box\psi_2)) \\ & \text{by rule (1)} \\ \Rightarrow & S, q \models \forall\Box((\varphi_1 \wedge \varphi_2) \rightarrow (\forall\Box\psi_1 \wedge \forall\Box\psi_2)) \\ & \text{by the boolean law } (p \rightarrow q) \wedge (r \rightarrow s) \Rightarrow (p \wedge r) \rightarrow (q \wedge s) \\ \Rightarrow & S, q \models \forall\Box((\varphi_1 \wedge \varphi_2) \rightarrow \forall\Box(\psi_1 \wedge \psi_2)) \\ & \text{by rule (1)} \end{array}$$

6. In order to show that $S, q \models \forall\Box(\varphi_1 \rightarrow \langle\langle A\rangle\rangle\Diamond\varphi_3)$, we need to show that for all q-computations λ and positions $i \geq 0$, if $S, \lambda[i] \models \varphi_1$, then $S, \lambda[i] \models \langle\langle A\rangle\rangle\Diamond\varphi_3$. Pick an arbitrary q-computation λ and fix it. Also pick an arbitrary position i and fix it. Assume that $S, \lambda[i] \models \varphi_1$. We need to show that $S, \lambda[i] \models \forall\Box\langle\langle A\rangle\rangle\Diamond\varphi_3$. By assumption we have that $S, q \models \forall\Box(\varphi_1 \rightarrow \forall\Box\varphi_1)$. Since $S, \lambda[i] \models \varphi_1$, we get $S, \lambda[i] \models \forall\Box\varphi_1$.

   By assumption, we also have $S, q \models \forall\Box\langle\langle A\rangle\rangle\Diamond\varphi_2$. Hence, there exists a strategy function F_A, such that for every computation

$\lambda_A \in out(\lambda[i], F_A)$, there exists a position $k \geq 0$ (depending on λ_A), such that $S, \lambda_A[k] \models \varphi_2$. Pick an arbitrary $\lambda' \in out(\lambda[i], F_A)$ and fix it. Let l be the position such that $S, \lambda'[l] \models \varphi_2$.

Note that λ' is a computation beginning with $\lambda[i]$. By the above paragraph, we have $S, \lambda[i] \models \forall\Box\varphi_1$. Therefore, we also obtain that $S, \lambda'[l] \models \varphi_1$.

Note that $\lambda'[0] = \lambda[i]$. Consider the q-computation λ_1 constructed by appending the computation λ' to $\lambda[0..i]$. In other words, let

$$\begin{array}{rcll} \lambda_1[j] & = & \lambda[j] & j \leq i \\ \lambda_1[j] & = & \lambda'[j-i] & j \geq i \end{array}$$

λ_1 is a q-computation. We have by assumption $S, q \models \forall\Box(\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3))$. Hence for each position $j \geq 0$, $S, \lambda_1[j] \models \varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$. In particular, $S, \lambda_1[i+l] \models \varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$. But $\lambda_1[i+l] = \lambda'[l]$. Hence $S, \lambda'[l] \models \varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$. By the above two paragraphs $S, \lambda'[l] \models \varphi_1$ and $S, \lambda'[l] \models \varphi_2$. Therefore, we get $S, \lambda'[l] \models \varphi_3$.

Therefore, given any computation $\lambda_A \in out(\lambda[i], F_A)$, there is a position k such that $S, \lambda_A[k] \models \varphi_3$. Hence, we obtain $S, \lambda[i] \models \langle\langle A \rangle\rangle \Diamond \varphi_3$ as required.

□

Game guarded command language. Instead of modeling protocols directly with ATS we use a more user-oriented notation: a guarded command language a la Dijkstra. The details about the syntax and semantics of this language (given in terms of ATS) can be found in (Henzinger et al., 2000). Intuitively, each player $a \in \Sigma$ disposes of a set of guarded commands of the form $guard_\xi \rightarrow update_\xi$. A computation-step is defined as follows: each player $a \in \Sigma$ chooses one of its commands whose boolean guard evaluates to true, and the next state is obtained by taking the conjunction of the effects of each update part of the commands selected by the players. Given an ATS described in terms of guarded commands, the finite state tool MOCHA automates the model-checking of ATL formulae over the specified ATS.

3.2. MODELING PROTOCOLS

Unlike the classical security protocols aiming at secrecy and authentication, optimistic contract-signing protocols usually consist of subprotocols that can be invoked at specified moments. Running a protocol

at a time not foreseen by the designer, may have unexpected side-effects. This may be used by a signer to gain an advantage over other signers. We believe that such concurrency issues are a major source of problems. Therefore, and since the high number of messages would create a serious state explosion, we only analyze the *structure* of the protocols and concentrate only on one single protocol instance. We now discuss our model in detail.

The protocol instance is modeled as an ATS and each protocol participant is modeled as a player in the ATS using the above introduced guarded command language. Besides, the branching aspect, another notable difference with more classical secrecy and authentication protocols, is that contract-signing protocols must be secure against malicious signer, rather than an external intruder. In order to model this we have two processes for each signer, one describing its honest behavior and the other the dishonest one. Communication is modeled using shared variables. Each protocol message is modeled using a boolean variable, initialized to *false* and set to *true* when it is sent. Sending of a message is modeled using guarded commands, where the guard depends on previously sent out messages. When modeling the honest behavior of a participant, we ensure that a given message is sent out only when specified by the protocol. In contrary, the guards are relaxed in the malicious version of the signer so that each message can be sent out, as soon as possible, *i.e.*, as soon as all messages needed to compose the given message are received. We do not explicitly model any cryptographic primitives, but only the fact that protocol messages can be sent out of order. Hence, a dishonest signer can send messages out of order and continue the protocol, even if it is supposed to stop. We manually decide which messages must be known in order to send some other message. Moreover, the communication between any two signers is assumed to be on private channels and we do not model the possibility to spy other channels. The trusted party is modeled to be always honest. For each pair of signers P_i and P_j, we use a proposition $P_i.S_{P_j}(m)$ to model that the signer P_i has obtained P_j's signature. Once $P_i.S_{P_j}(m)$ is set to true, it never changes it value, *i.e.*, we assume that P_i continues to hold P_j's signature once P_i obtains it. For each signer P_i, we also use a special proposition `Pi_stop` to model that the entity P_i has quit the protocol. Once `Pi_stop` has been set to true, P_i would not be able to change any of its variables as $\neg$`Pi_stop` is present in each of its guards.

As an example, consider the short extract of the modeling of the three-party GM protocol depicted in figure 1. In the extract, the integer variable `Pr_i_j_L` models the promises that P_i has sent to P_j, and `Pr_i_j_L = k` means that P_i has sent out up to k-level promises to P_j.

(For efficiency reasons, we use the logarithmic encoding of a ranged integer variable, rather than having one boolean variable for each level of promise). In the extract, the first rule of honest P_1 says that P_1 may quit the protocol, if it has not contacted the trusted party, and has neither received nor sent any promises. The corresponding modeling of dishonest behavior of P_1 states that P_1 may quit the protocol at any moment. The second rule of honest P_1 gives the exact condition when the first level promise has to be sent to P_2. The corresponding dishonest rule, merely requires that P_1 has not quit the protocol before sending the promise.

Extract of honest modeling of P_1 for the three-party GM protocol:

```
[] ~P1_stop & ~P1_contacted_T & Pr_1_3_L=0 & Pr_1_2_L=0 &
   ~( Pr_3_1_L>0 & Pr_2_1_L>0 ) -> P1_stop':=true
[] ~P1_stop & ~P1_contacted_T & Pr_1_3_L=0 & Pr_1_2_L=0 &
   Pr_2_1_L>0 & Pr_3_1_L>0 -> Pr_1_2_L':=1
```

The corresponding actions of a dishonest modeling:

```
[] ~P1_stop -> P1_stop':=true
[] ~P1_stop & Pr_1_2_L<1 -> Pr_1_2_L':=1
```

***Figure 1.*:** Extract of the three-party GM protocol modeling

As we are unable to verify parametric systems with MOCHA, we simplify our task and verify the protocols only for a given number n of signers. In order to avoid encoding each instance of the protocol using guarded commands, we have written a dedicated C++ program for each protocol which takes the number n of signers as a parameter and generates the protocol specification. Although our model is restricted with regard to several aspects, the model seems to be of interest as several unknown anomalies have been revealed.

3.3. MODELING PROPERTIES

The desired security guarantees are expressed using ATL. In this section we discuss the modeling of fairness and timeliness. The modeling for abuse-freeness shall be discussed when we present the analysis of GM in section 4. Consider an instance of the protocol with n signers, which we denote as $P_1, \ldots, P_n$. In the following, we assume that only one of the signers, say P_1 is honest, and the other dishonest signers are colluding to cheat the honest signer.

A protocol is fair for an honest P_i, if at the end of the protocol, either P_i receives signed contracts from all the other signers or it is not

possible for any other signer to obtain P_i's signed contract. One possible ATL formula, henceforth referred to as StratFair$_i$, for modeling this says that if any signer receives P_i's signed contract, then P_i has a strategy to get the signed contracts from other signers:

$$\text{StratFair}_i \equiv \forall\Box((\bigvee_{1\leq j\neq i\leq n} P_j.S_{P_i}(m)) \rightarrow \langle\langle P_i\rangle\rangle\Diamond(\bigwedge_{1\leq j\neq i\leq n} P_i.S_{P_j}(m))) \quad (1)$$

where $P_i.S_{P_j}(m)$ denotes that player P_i received P_j's signature on the contract text m.

It can of course be argued that P_i having a strategy to receive the signed contracts is not a sound modeling of fairness: P_i may have this strategy but if it is ignorant or if it mistakenly does not follow this strategy, then the protocol may end in an unfair state. Therefore one could require the following stronger property (henceforth referred to as StrongFair$_i$): in whatever way P_i resolves the remaining choices specified by the protocol, P_i receives all the signed contract.

$$\text{StrongFair}_i \equiv \forall\Box((\bigvee_{1\leq j\neq i\leq n} P_j.S_{P_i}(m)) \rightarrow \forall\Diamond(\bigwedge_{1\leq j\neq i\leq n} P_i.S_{P_j}(m))) \quad (2)$$

In the same vein, a third weaker formulation (henceforth referred to as WeakFair$_i$) only requires that there exists a path where P_i receives the signed contracts.

$$\text{WeakFair}_i \equiv \forall\Box((\bigvee_{1\leq j\neq i\leq n} P_j.S_{P_i}(m)) \rightarrow \exists\Diamond(\bigwedge_{1\leq j\neq i\leq n} P_i.S_{P_j}(m))) \quad (3)$$

We have that

$$\text{StrongFair}_i \Rightarrow \text{StratFair}_i \Rightarrow \text{WeakFair}_i.$$

In (Chadha et al., 2004), we concentrated on the last, weakest version of fairness. As we demonstrated that fairness is violated even in this weakest version, the other stronger versions are also violated.

One disadvantage with the above formulations of fairness is that Mocha does not provide counter-examples for arbitrary ATL formulae[2]. If a violation of fairness is found, we can use weaker formulae to discover scenarios leading up to the violation. An error trace usually speeds up the analysis considerably. Since Mocha does provide error traces for invariant violations, it is interesting to consider yet another formulation of the fairness property as an invariant. In (Shmatikov and Mitchell, 2002), invariants have already been used to approximate fairness. The idea is to consider only the states where an honest signer has

[2] Mocha does not give counter-examples, because in general, counter-examples cannot always be expressed as traces, even for CTL.

stopped the protocol. We know that once it has stopped the protocol, it cannot receive any new messages. Hence, in any state where P_i has stopped the protocol (modeled by setting a special proposition $P_i.stop$ to true), fairness would require P_i to have the complete contract if some $P_j, j \neq i$, has P_i's contract.

The invariant formulation of fairness for P_i tests only those states in which P_i has stopped the protocol, and is as follows.

$$\textsc{InvFair}_i \equiv \forall\Box(P_i.stop \rightarrow ((\bigvee_{1\leq j\neq i\leq n} P_j.S_{P_i}(m)) \rightarrow (\bigwedge_{1\leq j\neq i\leq n} P_i.S_{P_j}(m)))) \tag{4}$$

We denote this formulation of fairness as $\textsc{InvFair}_i$. Indeed, this formulation of fairness is weaker than the above formulations. In particular, we show that the formula $\textsc{WeakFair}_i$ implies $\textsc{InvFair}_i$. Before proceeding to the proof, please note that $\bigvee_{1\leq j\neq i\leq n} P_j.S_{P_i}(m)$ denotes that at least one signer $P_j, j \neq i$, has obtained P_i's contract. The formula $\bigwedge_{1\leq j\neq i\leq n} P_i.S_{P_j}(m)$ denotes that P_i has obtained contracts of all the other signers. Keeping this in mind, we use the following abbreviations for the rest of the section:

$$sig\_released_i \equiv \bigvee_{1\leq j\neq i\leq n} P_j.S_{P_i}(m)$$
$$sig\_obtained_i \equiv \bigwedge_{1\leq j\neq i\leq n} P_i.S_{P_j}(m)$$

In the proof, we shall also use the additional assumption (see section (3.2)) that once P_i stops the protocol, he cannot change his variables anymore. In particular, if S is the ATS that models the protocol for signers $P_1, \ldots, P_n$ and q_0 is any arbitrary initial state of S, we shall assume that for each $1 \leq i, j \leq n, i \neq j$:

$$S, q_0 \models \forall\Box((P_i.stop \wedge \neg P_i.S_{P_j}(m)) \rightarrow \forall\Box\neg P_i.S_{P_j}(m)) \tag{5}$$

This assumption can also be easily verified by Mocha, as a *sanity* check of the specification. Using the above assumption and proposition 1, we get the following proposition:

Proposition 2. If S is the ATS that models a contract signing protocol for signers $P_1, \ldots, P_n$ and q_0 is an arbitrary initial state of S, then for each $1 \leq i \leq n$ we have

$$S, q_0 \models \forall\Box((P_i.stop \wedge \neg sig\_obtained_i) \rightarrow \forall\Box\neg sig\_obtained_i) \tag{6}$$

Proof. For each $1 \leq j \leq n, i \neq j$, we have by assumption (5) given above that

$$S, q_0 \models \forall\Box((P_i.stop \wedge \neg P_i.S_{P_j}(m)) \rightarrow \forall\Box\neg P_i.S_{P_j}(m))$$

Hence, by applying rule (3) of proposition 1 $n-2$ times, we get

$$S, q_0 \models \forall\Box((P_i.stop \wedge (\bigvee_{1\leq j\neq i\leq n} \neg P_i.S_{P_j}(m))) \rightarrow \forall\Box \bigvee_{1\leq j\neq i\leq n} \neg P_i.S_{P_j}(m))$$

The result is now obtained by realizing that $\bigvee_{1\leq j\neq i\leq n} \neg P_i.S_{P_j}(m) \equiv \neg \bigwedge_{1\leq j\neq i\leq n} P_i.S_{P_j}(m) \equiv \neg sig\_obtained_i$. □

We are ready to prove that fairness in terms of the invariants is weaker than other formulations:

Theorem 1. Let S be an ATS that models a contract-signing protocol for signers $P_1, \ldots, P_n$ and q_0 be an arbitrary initial state of S. If $S, q_o \models \textsc{WeakFair}_i$, then $S, q_o \models \textsc{InvFair}_i$.

Proof. We shall prove the theorem by contraposition. Assume that $S, q_o \not\models \textsc{InvFair}_i$. We shall show that $S, q_o \not\models \textsc{WeakFair}_i$. Note that $S, q_o \not\models \textsc{InvFair}_i$ if and only if $S, q_o \models \neg\textsc{InvFair}_i$, and $S, q_o \not\models \textsc{WeakFair}_i$ if and only if $S, q_o \models \neg\textsc{WeakFair}_i$. Using the abbreviations $sig\_released_i$ and $sig\_obtained_i$, we have

$$\begin{aligned} \textsc{WeakFair}_i &\equiv \forall\Box(sig\_released_i \rightarrow \exists\Diamond sig\_obtained_i) \\ \textsc{InvFair}_i &\equiv \forall\Box(P_i.stop \rightarrow (sig\_released_i \rightarrow sig\_obtained_i)) \\ &\equiv \forall\Box((P_i.stop \wedge sig\_released_i) \rightarrow sig\_obtained_i) \end{aligned}$$

The last equivalence follows from the boolean law $p \rightarrow (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$. Hence, we have

$$\begin{aligned} \neg\textsc{InvFair}_i &\equiv \neg(\forall\Box((P_i.stop \wedge sig\_released_i) \rightarrow sig\_obtained_i)) \\ &\equiv \exists\Diamond\neg((P_i.stop \wedge sig\_released_i) \rightarrow sig\_obtained_i)) \\ &\equiv \exists\Diamond(P_i.stop \wedge sig\_released_i \wedge \neg sig\_obtained_i) \\ &\equiv \exists\Diamond((P_i.stop \wedge \neg sig\_obtained_i) \wedge sig\_released_i) \end{aligned}$$

The second equivalence follows from the duality of $\forall\Box$ and $\exists\Diamond$ ($\neg\forall\Box\varphi \equiv \exists\Diamond\neg\varphi$). The third equivalence follows from the boolean law $\neg(p \rightarrow q) \equiv (p \wedge \neg q)$. The last equivalence is just a rearrangement of formulas using commutativity of conjunction. Intuitively, the resulting formula for $\neg\textsc{InvFair}_i$ says that a protocol is not fair for the signer P_i, if there is a state in which some signer $P_j, j \neq i$, has P_i's contract ($sig\_released_i$) and P_i has stopped ($P_i.stop$) the protocol without P_i having the completed contract ($sig\_obtained_i$).

Now, by proposition 2, we have

$$S, q_o \models \forall\Box((P_i.stop \wedge \neg sig\_obtained_i) \rightarrow \forall\Box\neg sig\_obtained_i).$$

Also

$$S, q_0 \models \neg\textsc{InvFair}_i \equiv \exists\Diamond((P_i.stop \wedge \neg sig\_obtained_i) \wedge sig\_released_i).$$

Therefore by rule (4) of proposition 1 (with $A = \Sigma$), we obtain

$$\begin{aligned} S, q_0 \models\ & \exists\Diamond(\forall\Box\neg sig\_obtained_i \wedge sig\_released_i) \\ & \equiv \exists\Diamond(\neg(\exists\Diamond sig\_obtained_i) \wedge sig\_released_i) \\ & \equiv \exists\Diamond\neg(sig\_released_i \rightarrow \exists\Diamond sig\_obtained_i) \\ & \equiv \neg\forall\Box(sig\_released_i \rightarrow \exists\Diamond sig\_obtained_i) \\ & \equiv \neg(\textsc{WeakFair}_i) \end{aligned}$$

The first equivalence above follows from duality of $\forall\Box$ and $\exists\Diamond$. The second equivalence is a consequence of the boolean law $\neg q \wedge p \equiv \neg(p \rightarrow q)$. The third equivalence is again a consequence of the duality of $\forall\Box$ and $\exists\Diamond$, and the last equivalence is the definition of the formula $\textsc{WeakFair}_i$. Therefore, we have $S, q_0 \models \neg\textsc{WeakFair}_i$ as required. □

We can also show by example that the fairness in terms of invariant is strictly weaker than any of the above formulations. Consider the following naive two-party protocol where P_1 and P_2 directly exchange their signatures on the contractual text.

$$\begin{aligned} &1.\ P_1 \rightarrow P_2 :\ S_{P_1}(m) \\ &2.\ P_2 \rightarrow P_1 :\ S_{P_2}(m) \end{aligned}$$

Assuming that the protocol specifies that P_1 can only stop after it receives P_2's signature on the contract, the protocol would not violate the invariant $\textsc{InvFair}_i$. This is simply because P_1 will not stop until it has received P_2's contract. On the other hand, the protocol would violate all the other formulations of fairness. In order to see this consider the scenario in which a dishonest P_2 quits the protocol after receiving P_1's contract and without sending P_2's contract. So P_1 will be stuck forever waiting for P_2's contract, and would have no further possible moves. It can be easily seen that this scenario violates $\textsc{WeakFair}_i$: P_2 has P_1's signature, but P_1 cannot get P_2's signature.

Contract-signing protocols are often designed to prevent situations in which signers are waiting indefinitely. Informally, a protocol is said to be *timely* if it is always possible for each signer to have some recourse to prevent unbounded waiting. One possible formulation of timeliness, henceforth referred to as $\textsc{WeakTimely}_i$, is that a protocol is timely for a signer P_i, if for each reachable state there is a path in which P_i has terminated:

$$\textsc{WeakTimely}_i \equiv \forall\Box(\exists\Diamond P_i.stop)$$

The naive protocol above does not respect this property for P_1. As explained before, a dishonest P_2 could stop after having received the P_1's contract. In all paths from this state, P_1 will be stuck and cannot stop.

A stronger formulation of timeliness could require that in every possible state, P_i should have a strategy to get to a state in which it has stopped executing the protocol:

$$\textsc{StratTimely}_i \equiv \forall\Box(\langle\langle P_i\rangle\rangle\Diamond P_i.stop)$$

Theorem 1 showed that the formulation of fairness in terms of invariant is weaker than any of the other formulations of fairness. In presence of timeliness, however, we can show that the formulation of fairness in terms of invariants coincides with other formulations of fairness. In particular, we shall show in Theorem 2 that if a protocol satisfies $\textsc{WeakTimely}_i$ then it satisfies the formula $\textsc{WeakFair}_i$ if and only if it satisfies $\textsc{InvFair}_i$. In Theorem 3, we will show that if a protocol satisfies $\textsc{StratTimely}_i$ then it satisfies $\textsc{StratFair}_i$ if and only if it satisfies $\textsc{InvFair}_i$.

In the proofs, we shall also use the additional assumption (see section (3.2)) that once P_i obtains signature of P_j, then it continues to hold that signature. If S is the ATS that models the protocol for signers $P_1, \ldots, P_n$ and q_0 is any arbitrary initial state of S, we shall assume that for each $1 \leq i, j \leq n, i \neq j$:

$$S, q_0 \models \forall\Box(P_i.S_{P_j}(m) \rightarrow \forall\Box P_i.S_{P_j}(m)) \tag{7}$$

Once again, this assumption can also be easily verified by Mocha, as a *sanity* check of the specification. Using the above assumption and proposition 1, we get the following proposition:

Proposition 3. If S is the ATS that models a contract signing protocol for signers $P_1, \ldots, P_n$ and q_0 is an arbitrary initial state of S, then for each $1 \leq i \leq n$ we have

$$S, q_0 \models \forall\Box(sig\_released_i \rightarrow \forall\Box sig\_released_i) \tag{8}$$

Proof. For each $1 \leq j \leq n, i \neq j$, we have by the assumption above:

$$S, q_0 \models \forall\Box(P_j.S_{P_i}(m) \rightarrow \forall\Box P_j.S_{P_i}(m))$$

Hence, by applying rule (5) of proposition 1 $n-2$ times, we get

$$S, q_0 \models \forall\Box(\textstyle\bigwedge_{1\leq j\neq i\leq n} P_j.S_{P_i}(m) \rightarrow \forall\Box \bigwedge_{1\leq j\neq i\leq n} P_j.S_{P_i}(m))$$

Since $sig\_released_i \equiv \bigwedge_{1\leq j\neq i\leq n} P_j.S_{P_i}(m)$, we obtain the desired result. □

Theorem 2. Let S be an ATS that models a contract-signing protocol for signers $P_1, \ldots, P_n$ and q_0 be an arbitrary initial state of S. If $S, q_o \models \textsc{WeakTimely}_i$, then $S, q_o \models \textsc{WeakFair}_i$ if and only if $S, q_o \models \textsc{InvFair}_i$.

Proof. The $\Rightarrow$ direction of the proof follows directly from theorem 1 and does not require timeliness. For the $\Leftarrow$ direction, we need to prove that

$$S, q_o \models (\textsc{WeakTimely}_i \wedge \textsc{InvFair}_i) \Rightarrow S, q_0 \models \textsc{WeakFair}_i$$

We have that

$$\begin{aligned} & \textsc{WeakTimely}_i \wedge \textsc{InvFair}_i \\ \equiv\ & \forall\Box(\exists\Diamond P_i.stop) \wedge \forall\Box(P_i.stop \rightarrow (sig\_released_i \rightarrow sig\_obtained_i)) \\ \equiv\ & \forall\Box(\exists\Diamond P_i.stop) \wedge \forall\Box(sig\_released_i \rightarrow (P_i.stop \rightarrow sig\_obtained_i)) \end{aligned}$$

The latter equivalence is obtained from the boolean law $p \rightarrow (q \rightarrow r) \equiv q \rightarrow (p \rightarrow r)$.

We have also proved above in proposition 3 that

$$S, q_0 \models \forall\Box(sig\_released_i \rightarrow \forall\Box sig\_released_i).$$

Hence, we obtain as a result of rule (6) of proposition 1 (with $A = \Sigma$) that if $S, q_0 \models (\textsc{WeakTimely}_i \wedge \textsc{InvFair}_i)$, then

$$S, q_0 \models \forall\Box(sig\_released_i \rightarrow \exists\Diamond sig\_obtained_i) \equiv \textsc{WeakFair}_i$$

Hence we have the desired result. □

Theorem 3. Let S be an ATS that models a contract-signing protocol for signers $P_1, \ldots, P_n$ and q_0 be an arbitrary initial state of S. If $S, q_o \models \textsc{StratTimely}_i$, then $S, q_o \models \textsc{StratFair}_i$ if and only if $S, q_o \models \textsc{InvFair}_i$.

Proof. As in the previous proof, the $\Rightarrow$ direction of the proof follows directly from theorem 1 and does not require timeliness. For the $\Leftarrow$ direction, we need to prove that

$$S, q_o \models (\textsc{StratTimely}_i \wedge \textsc{InvFair}_i) \Rightarrow S, q_0 \models \textsc{StratFair}_i$$

We have that

$$\begin{aligned} & \textsc{StratTimely}_i \wedge \textsc{InvFair}_i \\ \equiv\ & \forall\Box(\langle\!\langle P_i \rangle\!\rangle \Diamond P_i.stop) \wedge \forall\Box(P_i.stop \rightarrow (sig\_released_i \rightarrow sig\_obtained_i)) \\ \equiv\ & \forall\Box(\langle\!\langle P_i \rangle\!\rangle \Diamond P_i.stop) \wedge \forall\Box(sig\_released_i \rightarrow (P_i.stop \rightarrow sig\_obtained_i)) \end{aligned}$$

The latter equivalence is obtained from the boolean law $p \rightarrow (q \rightarrow r) \equiv q \rightarrow (p \rightarrow r)$.

We have also proved above in proposition 3 that

$$S, q_0 \models \forall\Box(sig\_released_i \rightarrow \forall\Box sig\_released_i).$$

Hence, we obtain as a result of rule (6) of proposition 1 that if $S, q_0 \models (\textsc{StratTimely}_i \wedge \textsc{InvFair}_i)$, then

$$S, q_0 \models \forall\Box(sig\_released_i \rightarrow \langle\langle P_i \rangle\rangle \Diamond sig\_obtained_i) \equiv \textsc{StratFair}_i$$

Hence we have the desired result. □

Note that the formula $\textsc{StratTimely}_i$ is a stronger condition than $\textsc{WeakTimely}_i$, *i.e.*, if a protocol satisfies $\textsc{StratTimely}_i$ then it will also satisfy $\textsc{WeakTimely}_i$. Using theorems 2 and 3, we therefore obtain that if a protocol satisfies timeliness in terms of strategy, then the three forms of fairness $\textsc{StratFair}_i$, $\textsc{WeakFair}_i$ and $\textsc{InvFair}_i$ are all equivalent. In our analysis, both the protocols that we analyzed did indeed satisfy $\textsc{StratTimely}_i$. We then proceeded to check if the protocols satisfied $\textsc{InvFair}_i$. As we just noted above, this is equivalent to checking if the protocols satisfied $\textsc{StratFair}_i$ and $\textsc{WeakFair}_i$.

Please note that just as we had another version of fairness, i.e., $\textsc{StrongFair}_i$, we could posit a stronger version of timeliness: in whatever state, every execution finally leads to a state of the protocol, where an honest signer P_i has quit the protocol. Just as in Theorems 2 and 3, we could then show that if a protocol satisfies this strong version of timeliness, then a protocol satisfies $\textsc{StrongFair}_i$ if and only if it satisfies $\textsc{InvFair}_i$. In our model, however, an honest signer does not satisfy this strong version of timeliness. This is because in our model we allow an honest signer to idle in every possible state. For the same reason, $\textsc{StrongFair}_i$ does not hold in our model. We shall not consider these versions of fairness and timeliness in our analysis. We are now ready to present the results of analysis.

4. Analysis

We now discuss the results of our analysis of the two protocols. Our analysis did not reveal any anomalies in the BW protocol, and hence shall discuss this protocol briefly.

4.1. Analysis of the BW protocol

We have verified the BW protocol with two and up to five signers for timeliness and fairness. The model-checker Mocha did not detect any flaw in these cases. The protocol is not intended to be abuse-free, and hence we did not study the protocol for this property. In our opinion, we found that the design of the BW protocol is much simpler than the GM protocol. The decision to overturn an abort is based on the following argument: T overturns only if it can infer that no previous abort reply has been sent to a potentially honest principal. This seems to ensure the robustness of the protocol. Note that we only analyzed the structure of the protocol. Hence, our results only prove that the protocol is correct in the given model.

4.2. Analysis of the GM protocol

We now report in more detail on our analysis of the GM contract-signing protocol. The protocol has several peculiarities. The most notable one is that the protocol changes with the number of signers, *e.g.*, the protocol specification of P_1 differs when the value of n changes. The number of protocol messages increases considerably with the number of signers. For instance, if we have $n = 3$, the main protocol has 20 messages and there are 14 different recovery requests. When $n = 4$, the corresponding numbers are 41 and 36. Moreover, the protocol is not symmetric for the signers: the protocol specification for P_i is different from that for P_j, for all $i \neq j$. For instance, when $n = 4$, P_1 can launch 18 different recovery requests and P_4 only 2.

As mentioned in section 3, we have written a dedicated C++ program that takes the number of signers, n, as a parameter and generates the protocol specification. Our analysis revealed problems with fairness, when n is 4. Although, we did not discover any fairness problems when $n = 3$, we did find an amusing problem with abuse-freeness. We did not discover any problems with timeliness in the protocol. All these anomalies are novel and the protocol was believed to be secure since it was first published. We discuss our results in detail. The source codes of our analysis of both protocols are also available at the following website `http://www.lsv.ens-cachan.fr/~kremer/MPCS/`.

Fairness. We analyzed the GM protocol for 3 and 4 signers, for both fairness and timeliness. Our analysis did not reveal any flaws in timeliness. As discussed in section 3, in presence of timeliness fairness can be expressed as an invariant. The advantage of invariant checking is that Mocha outputs an error-trace whenever an invariant is violated. Moreover, invariant checking is more efficient.

We did not discover any problems with fairness when $n = 3$. The formulas representing fairness for P_1, P_2 and P_3, introduced in section 3, are validated by MOCHA. However, as we use a restricted model and consider single runs, we can only conclude that the protocol does not present any *structural* weakness for $n = 3$. Indeed, if we relax the assumption of the private channels, the anomaly presented by Shmatikov and Mitchell in (Shmatikov and Mitchell, 2002) on the two-party GJM protocol can be adapted to the multi-party version. In this scenario, a malicious signer eavesdrops on the channel between the honest signer and T, and succeeds in compromising fairness. With our present modeling, we do not find such flaws, as this requires to eavesdrop channels and to decompose messages. The fix proposed by Shmatikov and Mitchell applies to the multi-party protocol too. However, we should emphasize that the authors of the GM protocol require the channels to T to be private and hence this scenario does not represent a valid attack on the protocol.

We discovered several scenarios that compromised fairness when $n = 4$. The first scenario was discovered by hand, when we found an error in the proof of correctness given in the original paper (Garay and MacKenzie, 1999). A detailed analysis using MOCHA detected seven other scenarios. An analysis of these revealed that the proof also did not cover a case. In each scenario, an honest signer is cheated by the coalition of three malicious signers. These scenarios follow the general outline:

1. A dishonest signer contacts the trusted party, T, at the beginning of the protocol, gets an abort token, and dishonestly continues participating in the main protocol.

2. A second dishonest signer tries to recover at some later point. It does not succeed, but manages to put the honest signer in the list F_m. It dishonestly continues the main protocol.

3. The honest signer is forced to recover, but is not successful in getting the abort decision overturned since it is in the list F_m.

4. The third dishonest signer contacts T and manages to overturn the decision. Hence, while the honest signer does not get any signed contract, the honest signer's contract is obtained.

We only describe one of these scenarios in detail. The other attack scenarios can be found in Appendix A. In this scenario, P_1, P_3 and P_4 collude to cheat P_2. The scenario proceeds as follows:

– At the beginning of the protocol, P_3 aborts the protocol and T updates $S_m = \{3\}$. However, unlike specified by the protocol, dishonest P_3 continues the main protocol execution.

– As soon as P_1 receives the second level promise from P_2, it asks T to recover by sending

$$S_{P_1}(\{PCS_{P_2}((m,2),P_1,T), PCS_{P_3}((m,1),P_1,T), \\ PCS_{P_4}((m,1),P_1,T)\}, S_{P_1}((m,1))).$$

T refuses this request, answers with an abort message and updates $S_m = \{1,3\}$ and $F_m = \{2\}$. As P_3 did before, P_1 also continues the protocol.

– The main protocol is executed normally until signer P_2 reaches point 6.2. (see table 1) of the protocol with $a = 4$. At that point P_2 has sent out the set of message

$$\{PCS_{P_2}((m,1),P_1,T), PCS_{P_2}((m,2),P_1,T), \\ PCS_{P_2}((m,2),P_3,T), PCS_{P_2}((m,3),P_1,T), \\ PCS_{P_2}((m,3),P_3,T), PCS_{P_2}((m,3),P_4,T)\}$$

and has received the set of messages

$$\{PCS_{P_4}((m,1),P_2,T), PCS_{P_3}((m,1),P_2,T), \\ PCS_{P_1}((m,1),P_2,T), PCS_{P_1}((m,2),P_2,T), \\ PCS_{P_3}((m,3),P_2,T), PCS_{P_1}((m,3),P_2,T)\}.$$

P_2 is at position 6.2. with $a = 4$ and is waiting for 4-level promises from P_3 and P_4. P_3 and P_4 do not reply and P_2 is forced to send the following recovery request to T.

$$S_{P_2}(\{PCS_{P_1}((m,3),P_2,T), PCS_{P_3}((m,3),P_2,T), \\ PCS_{P_4}((m,1),P_2,T)\}, S_{P_2}((m,1)))$$

P_2 is in F_m, the tests in the protocol description (see table 3) indicate that T refuses the request, updates $S_m = \{1,2,3\}$ and replies with an abort message. F_m remains unchanged.

– P_4 launches a resolve request, sending

$$S_{P_4}(\{PCS_{P_1}((m,3),P_4,T), PCS_{P_2}((m,3),P_4,T), \\ PCS_{P_3}((m,3),P_4,T)\}, S_{P_4}((m,1))).$$

This request overturns the previous aborts, and hence violates fairness as T sends the signed contract back to P_4.

Such attacks are detected by MOCHA when checking the invariant representing fairness. A "cleaned up"[3] version of a sample error trace illustrating an attack compromising fairness for P_3 is shown in figure 2.

We also discovered that the protocol is unfair for signers P_1, P_2 and P_3 when $n = 4$. We give the other attacks in Appendix A. We did not find any scenario that compromised fairness for P_4. This is probably because the tests indicate that P_4 can never be added to F_m, when $n = 4$.

Threshold security. Our analysis revealed that the protocol is flawed for $n = 4$ signers when we have $n - 1 = 3$ dishonest parties. An interesting line of investigation is the threshold of the protocol security, i.e., how many dishonest signers are required to break fairness. While allowing only 1 malicious signer, MOCHA did not detect any anomaly when n is 4. However, MOCHA showed a peculiar scenario when we considered two dishonest signers when $n = 4$. In fact, the dishonest signers may bring the protocol to a state, where one of the honest signer is fooled, while the other honest signer obtains all other signatures. The amusing part of this scenario is that the dishonest signers themselves do not obtain all signatures, i.e., they can only decide to give an advantage to one of the participants while fooling the other one. This protocol execution should be considered as unfair, even if the dishonest signers do not obtain a direct advantage. They choose to give an advantage to one of the two honest participants. We believe that this gives an interesting insight into fairness, which cannot be observed in a two-party protocol. We describe one of these attacks in Appendix B.

Correcting the Garay-MacKenzie Protocol. In order to restore fairness in the Garay-MacKenzie protocol, we had to do major revisions in the recovery protocol. We were unsuccessful to restore fairness with minor changes, and we believe that this is because the meaning of the list $F(m)$ is not clear in the protocol. The central idea behind the revision is that T, when presented with a recovery request, overturns its abort decision if and only if T can infer dishonesty on the part of each of the signer that contacted T in the past. This is also the main idea behind the recovery protocol in (Baum-Waidner and Waidner, 2000).

The main protocol remains the same. Major changes are in the recovery protocol. The recovery messages are designed so that T can infer

[3] An error-trace shows a sequence of states, each described by the valuation of the variables; we used a script which transforms the output into a more readable form, showing only the variables which have changed with respect to the previous state.

```
T_Abort_Send_P1=0 T_Abort_Send_P2=0 T_Abort_Send_P3=0
T_Abort_Send_P4=0 T_F1=0 T_F2=0 T_F3=0 T_F4=0 T_Recovery_Send_P1=0
T_Recovery_Send_P2=0 T_Recovery_Send_P3=0 T_Recovery_Send_P4=0
T_Respond1=0 T_Respond2=0 T_Respond3=0 T_Respond4=0 T_S1=0 T_S2=0
T_S3=0 T_S4=0 T_Validated=0 P3_AbortToken=0 P3_Abort_Send=0
P3_Recovery_1_2_2=0 P3_Recovery_1_3_3=0 P3_Recovery_4_3_3=0
P3_Recovery_4_4_4=0 P3_Recovery_5_4_4=0 P3_S1=0 P3_S2=0 P3_S4=0
P3_contacted_T=0 P3_stop=0 Pr_3_1_L=0 Pr_3_2_L=0 Pr_3_4_L=0
P2_AbortToken=0 P2_Abort_Send=0 P2_Recovery_1_1_1=0
P2_Recovery_1_1_2=0 P2_Recovery_1_3_2=0 P2_Recovery_1_3_3=0
P2_Recovery_1_4_3=0 P2_Recovery_4_3_3=0 P2_Recovery_4_4_3=0
P2_Recovery_4_4_4=0 P2_Recovery_4_5_4=0 P2_Recovery_5_4_4=0
P2_Recovery_5_5_4=0 P2_S1=0 P2_S3=0 P2_S4=0 P2_contacted_T=0 P2_stop=0
Pr_2_1_L=0 Pr_2_3_L=0 Pr_2_4_L=0 P4_AbortToken=0 P4_Abort_Send=0
P4_Recovery_3_3_3=0 P4_Recovery_4_4_4=0 P4_S1=0 P4_S2=0 P4_S3=0
P4_contacted_T=0 P4_stop=0 Pr_4_1_L=0 Pr_4_2_L=0 Pr_4_3_L=0
P1_AbortToken=0 P1_Recovery_1_1_1=0 P1_Recovery_1_1_2=0
P1_Recovery_1_1_3=0 P1_Recovery_1_3_2=0 P1_Recovery_1_3_3=0
P1_Recovery_1_3_4=0 P1_Recovery_1_4_3=0 P1_Recovery_1_4_4=0
P1_Recovery_4_3_3=0 P1_Recovery_4_3_4=0 P1_Recovery_4_4_3=0
P1_Recovery_4_4_4=0 P1_Recovery_4_4_5=0 P1_Recovery_4_5_4=0
P1_Recovery_4_5_5=0 P1_Recovery_5_4_4=0 P1_Recovery_5_4_5=0
P1_Recovery_5_5_4=0 P1_S2=0 P1_S3=0 P1_S4=0 P1_contacted_T=0 P1_stop=0
Pr_1_2_L=0 Pr_1_3_L=0 Pr_1_4_L=0

Pr_4_3_L=1

Pr_3_1_L=1 Pr_3_2_L=1 Pr_2_3_L=2 Pr_1_3_L=2

Pr_3_1_L=3 Pr_3_2_L=3 Pr_2_3_L=3 Pr_4_3_L=4 Pr_1_3_L=3

Pr_3_4_L=3 Pr_4_2_L=1 Pr_1_3_L=4

Pr_3_1_L=4 Pr_3_2_L=4 Pr_2_3_L=4 P4_Abort_Send=1 Pr_1_2_L=2

T_Abort_Send_P4=1 T_Respond4=1 T_S4=1 Pr_3_4_L=4 P2_Recovery_1_3_2=1

T_Abort_Send_P2=1 T_F3=1 T_Respond2=1 T_S2=1 P3_Recovery_4_4_4=1
P3_contacted_T=1 Pr_2_1_L=4 Pr_4_1_L=4

T_Abort_Send_P3=1 T_Respond3=1 T_S3=1 P1_Recovery_4_4_4=1

T_Recovery_Send_P1=1 T_Respond1=1 T_Validated=1 P3_AbortToken=1
P3_stop=1

P1_S2=1 P1_S3=1 P1_S4=1
```

Figure 2.: Counter-example of fairness for P_3

Table 6 Revised GM multi-party contract-signing protocol—Abort for P_i

The first time T is contacted for contract m (either abort or recovery), T initializes $F(m)$ to $\emptyset$ and validated(m) to false.

1. $P_i \rightarrow$ T: $S_{P_i}(m, P_i, (P_1, \ldots, P_n), abort)$

if not validated(m) then

$S(m) = S(m) \cup \{i\}$
if $S(m) = \emptyset$ then T stores $S_T(S_{P_i}(m, P_i, (P_1, \ldots, P_n), abort))$
2. $T \rightarrow P_i$: $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort), S_T(m, S(m), abort))$

else (validated(m)=true)

3. $T \rightarrow P_i$: $\{S_{P_j}((m, k_j))\}_{j \in \{1,\ldots,n\} \setminus \{i\}}$
where k_j is the level of the promise from P_j that was converted to a universally-verifiable signature during the recovery protocol.

the promises that an *honest* signer would have sent when it launched the recovery protocol (note that a signer may have dishonestly sent other promises). For P_i to recover, it sends the message

$$S_{P_i}(\{PCS_{P_j}((m, k_j), P_i, T)\}_{j \in \{1,\ldots,n\} \setminus \{i\}}, S_{P_i}((m, 1)))$$

where k_j is computed as following:

1. If P_i runs the resolve protocol in step 5 of the main protocol (see table 1), then $k_j = 1$ for $j > i$ and $k_j = i - 1$ for $j < i$.
2. In step 6.2 of the main protocol, $k_j = a - 1$ for $1 < j \leq a - 1, j \neq i$ and $k_j = 1$ for $j > a - 1$.
3. In step 6.4 of the main protocol, $k_j = a - 1$ for $j < i$, $k_j = a$ for $i < j \leq a$ and $k_j = 1$ for $j > a$.
4. In step 7 of the main protocol, $k_j = n$ for all j.
5. In step 9 of the main protocol, $k_j = n$ for all $j < i$ and $k_j = n + 1$ for all $j > i$.

k_j may alternately be computed as:

- If $j < i$, k_j is the maximum level of promises received from all signers $P_{j'}$, with $j' < i$, *i.e.* the min-max of the promises from signers with lower index. (For example, if the maximum level of the promises received by P_4 from P_3 and P_2 was 6, and the maximum level received by P_4 from P_1 was 5, then $k_j = 5$ for $1 \leq j \leq 3$.)
- Let l be the maximum value l' such that P_i has l' level promises from P_j for all $i \leq j \leq l'$. If no such l' exists then let l be 0. If

Table 7 Revised GM multi-party contract-signing protocol—Recovery

The first time T is contacted for contract m (either abort or recovery), T initializes $F(m)$ to $\emptyset$ and validated(m) to false.

1. $P_i \rightarrow$ T: $S_{P_i}(\{PCS_{P_j}((m,k_j),P_i,T)\}_{j\in\{1,\ldots,n\}\setminus\{i\}}, S_{P_i}((m,1)))$

if $i \in S(m)$ then

T ignores the message

else if validated(m) then

2. $T \rightarrow P_i$: $\{S_{P_j}((m,k_j))\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$
where k_j is the level of the promise from P_j that was converted to a universally-verifiable signature.

else if $S(m) = \emptyset$ then

validated(m):=true
3. $T \rightarrow P_i$: $\{S_{P_j}((m,k_j))\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$

else (validated(m)=false $\wedge$ $S(m) \neq \emptyset$)

a) If there is some $p < i$ in $S(m)$ such that $k_p \leq h_p(m)$, or if there is some $p > i$ in $S(m)$ such that $k_p \leq l_p(m)$, then T sends back the stored abort $S_T(S_{P_j}(m, P_j, (P_1, \ldots, P_n), abort))$ to P_i. T adds i to $S(m)$, and computes $h_i(m)$ and $l_i(m)$ as follows

$(h_i(m), l_i(m)) = (k_{i+1}, 0)$,	if $i = 1$ (intuitively, P_1 has contacted T in either step 6.2 of the main protocol with $a = k_{i+1} + 1$ or in step 7 of the main protocol),
$= (0, i)$,	if $1 < i$ and $k_{i-1} = i-1$ (intuitively, P_i has contacted T in step 5 of the main protocol),
$= (k_{i-1}, k_{i-1})$,	if $1 < i < n, i \leq k_{i-1} \leq n$ and $k_{i+1} \leq k_{i-1}$ (intuitively, P_i has contacted T in step 6.2 of the main protocol with $a = k_{i-1} + 1$),
$= (k_{i-1}, k_{i-1} + 1)$,	if $1 < i < n, i \leq k_{i-1} < n$ and $k_{i+1} > k_{i-1}$ (intuitively, P_i has contacted T in step 6.4 of the main protocol with $a = k_{i-1} + 1$),
$= (n, n)$,	if $1 < i < n$ and $k_{i-1} = k_{i+1} = n$. (intuitively, P_i has contacted T in step 7 of the main protocol).
$= (n+1, n+1)$,	if $1 < i < n, k_{i-1} = n$ and $k_{i+1} = n+1$. (intuitively, P_i has contacted T in step 9 of the protocol).
$= (0, n+1)$,	if $i = n$ and $k_{i-1} = n$. (intuitively, P_n has contacted T in step 9 of the main protocol).

b) Otherwise, T sends $\{S_{P_j}((m,k_j))\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$ to P_i, stores all the signatures, and sets validated(m) to true.

$l = 0$, then let $k_j = 1$ for all $j > i$. If $l \neq 0$, then let $k_j = l$ for all $i \leq j \leq l$ and $k_j = 1$ for all $j > l$. (For example, if P_2 has received level 1 promise from P_6, level 5 and 1 promises from P_5, level 5, 4 and 1 promises from P_4, and level 4, 3 and 1 promises from P_3 then $k_6 = 1, k_5 = 1, k_4 = 4, k_3 = 4$.)

T maintains the set $S(m)$ of indices of signers that contacted T in the past and received an abort token. For each signer P_i in the set $S(m)$, T also maintains two integer variables $h_i(m)$ and $l_i(m)$. Intuitively, h_i is the highest level promise an honest P_i could have sent to any higher indexed signer before it contacted T. l_i is the highest level promise an honest P_i could have sent to a lower indexed signer before it contacted T. The protocol for T works as follows:

- If T ever replies with a signed contract for m, then T responds with the contract for any further request.
- If the first request to T is a resolve request, then T sends back a signed contract.
- If the first request is an abort request, then T aborts the contract. T may overturn this decision in the future if it can deduce that all the signers in $S(m)$ have behaved dishonestly. T deduces that a signer P_i in $S(m)$ is dishonest when contacted by P_j if
  1. $j > i$ and P_j presents to T a k-level promise from P_i such that $k > h_i(m)$, or
  2. $j < i$ and P_j presents to T a k-level promise from P_i such that $k > l_i(m)$.

We describe the abort and recovery protocols in detail in table 6, respectively 7.

We analyzed the revised protocol for both 3 and 4 signers and MOCHA did not detect any errors in the revised protocol. Please note that this should not be construed as proof of correctness since we are using a restricted communication model and are modeling a single run.

Abuse-freeness. We now describe the anomaly that we discovered for $n = 3$ signers in the GM protocol. The anomaly exploits the fact that when T replies with an abort decision, it also signs the list $S(m)$ of the signers who have received an abort from T. Recall that an optimistic signer (Chadha et al., 2003) is one that prefers to wait for "some time" before contacting the trusted party. Following (Chadha et al., 2003), we say that a protocol is abuse-free for a signer P_i if the protocol does not provide *provable advantage* to the remaining signers. A coalition of

signers is said to have provable advantage against P_i at a point in the protocol if (*i*) they have a strategy to abort the contract against an optimistic P_i, (*ii*) they have a strategy to get optimistic P_i's contract, and (*iii*) they can prove to an outside challenger, Charlie, that P_i is participating in the protocol.

Now consider the protocol instance with three signers P_1, P_2 and P_3. Assume that P_3 is optimistic and P_1 and P_2 are colluding to cheat P_3. P_3 starts the protocol by sending its level 1 promises to P_1 and P_2, and, being optimistic, waits for level 2 promises from them. P_2 on receiving this sends its level 1 promise to P_1, and then sends an abort request to T which aborts the protocol. Now, P_1 has received level 1 promises from P_2 and P_3. Using these first level promises, P_1 sends a recovery request to T. Note that, in the protocol, P_1 is never allowed to abort and T would not accept an abort request from P_1. P_1's recovery request is refused and T sends

$$S_T(S_{P_2}(m, P_2, (P_1, P_2, P_3), abort))$$

and

$$S_T(m, S(m) = \{1, 2\}, abort)$$

At this point, we make the following observations:

- the abort reply contains the set $S(m) = \{1, 2\}$ and is different from the one P_2 received,
- if P_1 receives an abort reply from T, it is always the answer to a recovery request,
- a recovery request always includes a promise from each signer which is verified by T.

From these remarks, we can conclude that if P_1 shows the abort reply to Charlie, then Charlie will be convinced that P_3 has started the protocol even though Charlie is unable to verify the PCS from P_3. In other words, we can say that T has verified the PCS for Charlie. Also note that at this point P_1 and P_2 can force the exchange to abort by simply quitting the protocol: P_3 has no promises from P_1 and P_2. P_1 and P_2 can also force a successful completion of the contract exchange by simply (dishonestly) engaging P_3 in the main protocol. Hence the protocol is not abuse-free for P_3.

This vulnerability can be easily addressed by excluding the set $S(m)$ from the abort reply. In this case, the abort messages from P_3 and P_2 are exactly similar and can be obtained by P_2 without P_3's participation. Hence, an abort reply does not prove P_3's participation in the protocol. This rather amusing scenario illustrates that sometimes additional

information may be harmful. While explicitness is often considered a good engineering practice (and we do not attempt to criticize such thumb rules), care should be taken when applying these principles. In personal communication with the authors of the protocol, they propose a different fix in letting P_1 abort the protocol rather than just quitting. If P_1 is allowed to abort the protocol, T's abort reply does not imply anymore that P_3 is participating: in contrast to the recovery request, when sending and abort request, T does not have to verify a PCS generated by P_3.

Abuse-freeness for P_3 is naturally expressed in ATL as follows:

$$\begin{array}{l}\neg\exists\Diamond(T.\,send(abort)\ to\ P_1\wedge\\ \quad\langle\langle P_1,P_2\rangle\rangle\Box(\neg P_3.S_{P_1}(m)\vee\neg P_3.S_{P_2}(m))\wedge\\ \quad\langle\langle P_1,P_2\rangle\rangle\Box(P_3.stop\rightarrow(P_1.S_{P_3}(m)\wedge P_2.S_{P_3}(m)))\\ )\end{array}$$

The boolean variable $T.\,send(abort)\ to\ P_1$ is set to true when P_1 receives the the abort token $S_T(S_{P_2}(m,P_2,(P_1,P_2,P_3),abort))$ with $S(m)=\{1,2\}$. As discussed before, this serves as a proof of P_3's participation. The variables $P_i.S_{P_j}(m)$ reflect that player i has received player j's signature on the contract. More precisely, the formula requires that it is not possible to reach a point where

1. P_1 and P_2 can prove to Charlie that the protocol was started by P_3 ($T.\,send(abort)\ to\ P_1$ is true),
2. P_1 and P_2 have a strategy to choose an unsuccessful outcome, *i.e.*, P_3 cannot get some signer's contract ($\langle\langle P_1,P_2\rangle\rangle\Box(\neg P_3.S_{P_1}(m)\vee\neg P_3.S_{P_2}(m))$ is true), and
3. P_1 and P_2 have a strategy to choose a successful outcome, *i.e.*, when honest P_3 stops, the dishonest players must have obtained P_3's contract ($\langle\langle P_1,P_2\rangle\rangle\Box(P_3.stop\rightarrow(P_1.S_{P_3}(m)\wedge P_2.S_{P_3}(m))$ is true).

The requirement of P_3 stopping in condition 3 is to prevent it from idling forever. Even though as discussed before, the protocol is not abuse-free if P_3 is optimistic, the above formula is validated if P_3 is honest. An honest P_3 may contact T non-deterministically as permitted by the protocol. Indeed, in the scenario discussed above, an honest P_3 could prevent P_1 and P_2 from getting P_3's signature if it contacts T. Therefore, in order to capture the scenario described above, we needed to model optimistic signers.

Following (Chadha et al., 2003), we implement an optimistic signer by adding *signals* that the signer uses to decide when to quit waiting

for messages from other signers and contact T. P_3 uses 3 signals for this: one to decide when to ask T to abort and 2 to decide when to contact T for the two recovery protocols that P_3 can launch. These signals are controlled by a new player, $P3TimeOuts$, that is added to the model.

The decision to abort is modeled by 2 boolean variables:

$$setTimeOutAbort \text{ and } expiredTimeOutAbort.$$

While P_3 changes the value of the variable $setTimeOutAbort$, the variable $expiredTimeOutAbort$ is changed by $P3TimeOuts$. When P_3 sends level 1 promise to P_1 and P_2, it sets the value of $setTimeOutAbort$ to $true$, and then waits for level 2 promises from them. $P3TimeOuts$ may set $expiredTimeOutAbort$ to $true$ once $setTimeOutAbort$ is set to $true$ by P_3. If the promises arrive before $expiredTimeOutAbort$ is $true$, then P_3 continues with the main protocol, otherwise P_3 may contact T with an abort request. The decision to send recovery requests are modeled similarly.

Following (Chadha et al., 2003), abuse-freeness is modeled by having a coalition of $P3TimeOuts$, P_1 and P_2. This coalition can choose a sufficiently "long time" to keep P_3 from contacting T, while allowing P_1 and P_2 to schedule its messages in order to get the desired result. Abuse-freeness can then be expressed as

$$\begin{array}{l}\neg\exists\Diamond(T.\,send(abort)\ to\ P_1\wedge \\ \quad \langle\!\langle P_1, P_2, P3TimeOuts\rangle\!\rangle\Box(\neg P_3.S_{P_1}(m) \vee \neg P_3.S_{P_2}(m))\wedge \\ \quad \langle\!\langle P_1, P_2, P3TimeOuts\rangle\!\rangle\Box(P_3.stop \rightarrow (P_1.S_{P_3}(m) \wedge P_2.S_{P_3}(m))) \\ )\end{array}$$

Please note that even an optimistic P_3 should eventually be allowed to contact T, otherwise P_3 may be stuck forever. Hence, $P3TimeOuts$ must eventually set $expiredTimeOutAbort$ and other signals to $true$. Ideally, this should be set non-deterministically. However, ensuring that a variable changes its value in MOCHA slows down verification considerably. In order to make the verification feasible, we put a maximum limit, $tick$, on the number of computation steps after which the value must change, and vary this limit manually. Please note that the signals may change before this limit is reached. This modeling is sound in the sense that if the formula is violated for some value of $tick$ then abuse-freeness must be violated: P_3 just needs to wait for sufficiently "long time" to allow P_1 and P_2 to schedule its messages. Indeed, the above property is violated when $tick$ is set to 3 giving us the attack on abuse-freeness. As expected, if we drop the player $P3TimeOuts$ in the formula, then the property is not violated: P_1 and P_2 are not able to schedule their messages ahead of P_3.

However, if P_3 is not optimistic, the above given formulation of abuse-freeness is not violated. In a non-optimistic setting, the vulnerability can be detected by weakening the third requirement of the formula in order to say that there is a trace in which P_3 does not contact T, and P_1 and P_2 get P_3's signed contract. The ATL formula which captures this is

$$\begin{array}{l}\neg\exists\Diamond(T.\,send(abort)\ to\ P_1\wedge \\ \quad \langle\langle P_1,P_2\rangle\rangle\Box(\neg P_3.S_{P_1}(m)\vee\neg P_3.S_{P_2}(m))\wedge \\ \quad \exists\Box(P_3.stop\rightarrow(P_1.S_{P_3}(m)\wedge P_2.S_{P_3}(m))) \\ )\end{array}$$

This property is violated even in the original non-optimistic model. MOCHA also detected the violation of a stronger version of abuse-freeness proposed in (Kremer and Raskin, 2002). This formulation requires that as soon as P_2 and P_3 can prove to Charlie that P_1 started the protocol, then they may not achieve an unsuccessful outcome anymore.

5. Conclusions and Future Work

We have studied two multi-party contract-signing protocols (Garay and MacKenzie, 1999; Baum-Waidner and Waidner, 2000) using a finite-state tool, MOCHA, that allows specification of properties in a branching-time temporal logic with game semantics. In order to make this analysis feasible, we model single runs and assume a restricted communication model. Our analysis did not find any errors in the BW protocol (Baum-Waidner and Waidner, 2000). We did encounter problems with fairness in the case of four signers in the GM protocol (Garay and MacKenzie, 1999). It appears that fairness cannot be restored without completely rewriting the subprotocols. The revised subprotocols are inspired by the BW protocol. We also discovered a rather amusing problem with abuse-freeness in the GM protocol with three signers that occurs because abort messages from the trusted party reveal who have contacted it in the past. This problem is easily addressed by ensuring that the trusted party does not send this extra information. We had to implement optimistic signers to demonstrate this problem using MOCHA. Overturning of abort decisions were important features of multi-party contract-signing that were not present in 2-party protocols. In particular, T's decision when to overturn was at the origin of the flaws in the GM protocol. However, it remains to be investigated formally whether this is a necessary feature to maintain fairness, i.e., whether in any contract-signing protocol for $n \geq 3$ signers there are scenarios in which T must overturn the abort decision.

We modeled single runs, and used a restricted communication and cryptographic model for our analysis. Previous work on two-party contract signing protocols has shown that they are prone to error in a more general setting. For example, in (Chadha et al., 2001; Gürgens and Rudolph, 2003), the authors exhibit problems with fairness when multiple sessions are involved, and in (Backes et al., 2003), the authors exhibit errors when black-box cryptography is replaced by provably secure cryptographic signature schemes. We plan to verify the protocols without fixing the number of signers. One major challenge in such a parametric verification is that the protocol descriptions change fundamentally with the number of signers in that the protocol for n signers is not merely putting n identical processes in parallel. We hope to prove the correctness of these protocols in a more general setting which accounts for cryptography, multiple concurrent sessions, and relaxes the communication model. We plan to use theorem-provers such as Isabelle (Nipkow et al., 2002) and PVS (Crow et al., 1995) to achieve this. Proofs might also involve, at least partially, abstraction techniques such as those used by Das and Dill (Das and Dill, 2001) in their work.

Subsequent to our work, independent work by A. Mukhamedov and M. Ryan (Mukhamedov and Ryan, 2005) has found that fairness fails even for the revised protocol for 5 signers (which does not contradict any of our results). Moreover, they have an informal argument to show that no resolve protocol can fix the problem.

References

Alur, R., T. A. Henzinger, and O. Kupferman: 1997, 'Alternating-time temporal logic'. In: *38th Annual Symposium on Foundations of Computer Science (FOCS '97)*. pp. 100–109.

Asokan, N., M. Schunter, and M. Waidner: 1997, 'Optimistic Protocols for Fair Exchange'. In: *4th ACM Conference on Computer and Communications Security*. Zurich, Switzerland, pp. 7–17.

Backes, M., B. Pfitzmann, and M. Waidner: 2003, 'Reactively secure signature schemes'. In: *6th Information Security Conference (ISC)*, Vol. 2851 of *Lecture Notes in Computer Science*. pp. 84–95.

Baum-Waidner, B. and M. Waidner: 2000, 'Round-Optimal and Abuse Free Optimistic Multi-party Contract Signing'. In: *Automata, Languages and Programming — ICALP 2000*, Vol. 1853 of *Lecture Notes in Computer Science*. Geneva, Switzerland, pp. 524–535.

Burk, H. and A. Pfitzmann: 1990, 'Value exchange systems enabling security and unobservability'. In: *Computers and Security, 9(8):715–721*.

Chadha, R., M. Kanovich, and A. Scedrov: 2001, 'Inductive methods and contract-signing protocols'. In: *8th ACM Conference on Computer and Communications Security*. Philadelphia, PA, USA, pp. 176–185.

Chadha, R., S. Kremer, and A. Scedrov: 2004, 'Formal Analysis of Multi-Party Fair Exchange Protocols'. In: *17th IEEE Computer Security Foundations Workshop*. Asilomar, CA, USA, pp. 266–279.

Chadha, R., J. C. Mitchell, A. Scedrov, and V. Shmatikov: 2003, 'Contract signing, optimism, and advantage'. In: R. M. Amadio and D. Lugiez (eds.): *CONCUR 2003 — Concurrency Theory*, Vol. 2761 of *Lecture Notes in Computer Science*. pp. 361–377.

Chadha, R., J. C. Mitchell, A. Scedrov, and V. Shmatikov: 2005, 'Contract signing, optimism and advantage'. *Journal of Logic and Algebraic Programming (Special issue on Modeling and Verification of Cryptographic Protocols)* **64**(2), 189–218.

Crow, J., S. Owre, J. Rushby, N. Shankar, , and M. Srivas: 1995, 'A Tutorial Introduction to PVS'. In: *Workshop on Industrial-Strength Formal Specification Techniques*. Boca Raton, Florida.

Das, S. and D. L. Dill: 2001, 'Successive approximation of abstract transition relations'. In: *Sixteenth Annual IEEE Symposium on Logic in Computer Science (LICS 01)*. pp. 51–60.

Even, S. and Y. Yacobi: 1980, 'Relations among Public Key Signature Systems'. Technical Report 175, Technion, Haifa, Israel.

Garay, J. A., M. Jakobsson, and P. D. MacKenzie: 1999, 'Abuse-Free Optimistic Contract Signing'. In: M. J. Wiener (ed.): *Advances in Cryptology—Crypto 1999*, Vol. 1666 of *Lecture Notes in Computer Science*. pp. 449–466.

Garay, J. A. and P. D. MacKenzie: 1999, 'Abuse-Free Multi-party Contract Signing'. In: P. Jayanti (ed.): *International Symposium on Distributed Computing*, Vol. 1693 of *Lecture Notes in Computer Science*. Bratislava, Slavak Republic, pp. 151–165.

Gürgens, S. and C. Rudolph: 2003, 'Security Analysis of (Un-) Fair Non-repudiation Protocols'. In: A. E. Abdallah, P. Ryan, and S. A. Schneider (eds.): *Formal Aspects of Security*, Vol. 2629 of *Lecture Notes in Computer Science*. London, UK, pp. 97–114.

Henzinger, T. A., R. Manjumdar, F. Y. Mang, and J.-F. Raskin: 2000, 'Abstract Interpretation of Game Properties'. In: J. Palsberg (ed.): *SAS 2000: Intertional Symposium on Static Analysis*, Vol. 1824 of *Lecture Notes in Computer Science*. Santa Barbara, USA, pp. 220–239.

Kremer, S. and J.-F. Raskin: 2002, 'Game Analysis of Abuse-free Contract Signing'. In: *15th IEEE Computer Security Foundations Workshop*. Cape Breton, Canada.

Mukhamedov, A. and M. D. Ryan: 2005, 'Personal communication'.

Nipkow, T., L. C. Paulson, and M. Wenzel: 2002, *sabelle/HOL - A Proof Assistant for Higher-Order Logic*, Vol. 2283 of *Lecture Notes in Computer Science*. Springer.

Shmatikov, V. and J. Mitchell: 2002, 'Finite-state Analysis of Two Contract Signing Protocols'. *Theoretical Computer Science, special issue on Theoretical Foundations of Security Analysis and Design* **283**(2), 419–450.

Appendix

A. Attacks on the GM protocol

We now briefly sketch each of the attack scenarios we discovered on the GM protocol.

Attacks against P_1. In a first attack against P_1, P_4 succeeds in getting P_1's signature on the contract, while P_1 does not get P_4's signature.

1. At the beginning P_3 aborts, but continues the protocol. We have that $S_m = \{3\}$ and $F_m = \emptyset$.

2. As soon as possible, P_2 recovers by sending

$$S_{P_2}(\{PCS_{P_1}((m,2),P_2,T), PCS_{P_3}((m,1),P_2,T), \\ PCS_{P_4}((m,1),P_2,T)\}, S_{P_2}((m,1)))$$

   but receives an abort reply from T. We have that $S_m = \{3,2\}$ and $F_m = \{1\}$.

3. P_1 is forced to recover by sending

$$S_{P_1}(\{PCS_{P_2}((m,3),P_1,T), PCS_{P_3}((m,3),P_1,T), \\ PCS_{P_4}((m,1),P_1,T)\}, S_{P_1}((m,1)))$$

   but receives an abort reply from T. We have that $S_m = \{3,2,1\}$ and $F_m = \{1\}$.

4. P_4 recovers by sending

$$S_{P_4}(\{PCS_{P_1}((m,3),P_4,T), PCS_{P_2}((m,3),P_4,T), \\ PCS_{P_3}((m,3),P_4,T)\}, S_{P_4}((m,1)))$$

   and succeeds in overturning the abort decision.

In a second attack against P_1, it is P_2 who succeeds in receiving P_1's signature. Note that the attack is rather different from the previous one, as P_3 is added to F_m in this attack.

1. At the beginning P_4 aborts, but continues the protocol. We have that $S_m = \{4\}$ and $F_m = \emptyset$.

2. As soon as possible, P_2 recovers by sending

$$S_{P_2}(\{PCS_{P_1}((m,3),P_2,T), PCS_{P_3}((m,3),P_2,T), \\ PCS_{P_4}((m,1),P_2,T)\}, S_{P_2}((m,1)))$$

   but receives an abort reply from T. We have that $S_m = \{4,2\}$ and $F_m = \{3,1\}$.

3. P_1 is forced to recover by sending

$$S_{P_1}(\{PCS_{P_2}((m,4),P_1,T),PCS_{P_3}((m,4),P_1,T),\\ PCS_{P_4}((m,4),P_1,T)\},S_{P_1}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,2,1\}$ and $F_m = \{3,1\}$.

4. P_3 recovers by sending

$$S_{P_3}(\{PCS_{P_1}((m,4),P_3,T),PCS_{P_2}((m,4),P_3,T),\\ PCS_{P_4}((m,5),P_3,T)\},S_{P_3}((m,1)))$$

and succeeds in overturning the abort decision.

There exists also a third attack against P_1. This time, it is P_2 who succeeds in getting P_1's signature.

1. At the beginning P_4 aborts, but continues the protocol. We have that $S_m = \{4\}$ and $F_m = \emptyset$.

2. As soon as possible, P_3 recovers by sending

$$S_{P_3}(\{PCS_{P_1}((m,3),P_3,T),PCS_{P_2}((m,3),P_3,T),\\ PCS_{P_4}((m,3),P_3,T)\},S_{P_3}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,3\}$ and $F_m = \{2,1\}$.

3. P_1 is forced to recover by sending

$$S_{P_1}(\{PCS_{P_2}((m,4),P_1,T),PCS_{P_3}((m,4),P_1,T),\\ PCS_{P_4}((m,4),P_1,T)\},S_{P_1}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,3,1\}$ and $F_m = \{2,1\}$.

4. P_2 recovers by sending

$$S_{P_2}(\{PCS_{P_1}((m,4),P_2,T),PCS_{P_3}((m,5),P_2,T),\\ PCS_{P_4}((m,5),P_2,T)\},S_{P_2}((m,1)))$$

and succeeds in overturning the abort decision.

Attacks against P_2. In addition to the attack on P_2 which we described in details above, there exists an attack where P_3 receives P_2's signature on the contract.

1. At the beginning P_4 aborts, but continues the protocol. We have that $S_m = \{4\}$ and $F_m = \emptyset$.

2. As soon as possible, P_1 recovers by sending

$$S_{P_1}(\{PCS_{P_2}((m,2),P_1,T), PCS_{P_3}((m,3),P_1,T), \\ PCS_{P_4}((m,1),P_1,T)\}, S_{P_1}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,1\}$ and $F_m = \{2\}$.

3. P_2 is forced to recover by sending

$$S_{P_2}(\{PCS_{P_1}((m,4),P_2,T), PCS_{P_3}((m,4),P_2,T), \\ PCS_{P_4}((m,4),P_2,T)\}, S_{P_2}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,2,1\}$ and $F_m = \{2\}$.

4. P_3 recovers by sending

$$S_{P_3}(\{PCS_{P_1}((m,4),P_3,T), PCS_{P_2}((m,4),P_3,T), \\ PCS_{P_4}((m,4),P_3,T)\}, S_{P_3}((m,1)))$$

and succeeds in overturning the abort decision.

There exists a third attack against P_2. This attack is similar to the previous one, in the sense that it is P_3 who obtains P_2's signature. However, the "attack technique" slightly differs, since here P_3 is in F_m, while this is not the case in the previous attack.

1. At the beginning P_4 aborts, but continues the protocol. We have that $S_m = \{4\}$ and $F_m = \emptyset$.

2. As soon as possible, P_1 recovers by sending

$$S_{P_1}(\{PCS_{P_2}((m,3),P_1,T), PCS_{P_3}((m,3),P_1,T), \\ PCS_{P_4}((m,1),P_1,T)\}, S_{P_1}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,1\}$ and $F_m = \{3,2\}$.

3. P_2 is forced to recover by sending

$$S_{P_2}(\{PCS_{P_1}((m,4),P_2,T), PCS_{P_3}((m,4),P_2,T), \\ PCS_{P_4}((m,4),P_2,T)\}, S_{P_2}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,2,1\}$ and $F_m = \{3,2\}$.

4. P_3 recovers by sending

$$S_{P_3}(\{PCS_{P_1}((m,4),P_3,T), PCS_{P_2}((m,4),P_3,T), \\ PCS_{P_4}((m,5),P_3,T)\}, S_{P_3}((m,1)))$$

and succeeds in overturning the abort decision.

Attacks against P_3. We also discovered two attacks against P_3. In the first attack, P_2 is able to obtain P_3's signature on the contract test.

1. At the beginning P_4 aborts, but continues the protocol. We have that $S_m = \{4\}$ and $F_m = \emptyset$.

2. As soon as possible, P_1 recovers by sending

$$S_{P_1}(\{PCS_{P_2}((m,2),P_1,T), PCS_{P_3}((m,3),P_1,T), \\ PCS_{P_4}((m,1),P_1,T)\}, S_{P_1}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,1\}$ and $F_m = \{3\}$.

3. P_3 is forced to recover by sending

$$S_{P_3}(\{PCS_{P_1}((m,4),P_3,T), PCS_{P_2}((m,4),P_3,T), \\ PCS_{P_4}((m,4),P_3,T)\}, S_{P_3}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,3,1\}$ and $F_m = \{3,2\}$.

4. P_2 recovers by sending

$$S_{P_2}(\{PCS_{P_1}((m,4),P_2,T), PCS_{P_3}((m,4),P_2,T), \\ PCS_{P_4}((m,4),P_2,T)\}, S_{P_2}((m,1)))$$

and succeeds in overturning the abort decision.

In the second attack on P_3, it is P_1 who succeeds in overturning the abort decision and getting P_3's signed contract.

1. At the beginning P_4 aborts, but continues the protocol. We have that $S_m = \{4\}$ and $F_m = \emptyset$.

2. As soon as possible, P_2 recovers by sending

$$S_{P_2}(\{PCS_{P_1}((m,2),P_2,T), PCS_{P_3}((m,3),P_2,T), \\ PCS_{P_4}((m,1),P_2,T)\}, S_{P_2}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,2\}$ and $F_m = \{3\}$.

3. P_3 is forced to recover by sending

$$S_{P_3}(\{PCS_{P_1}((m,4),P_3,T), PCS_{P_2}((m,4),P_3,T), \\ PCS_{P_4}((m,4),P_3,T)\}, S_{P_3}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{4,3,1\}$ and $F_m = \{3\}$.

4. P_1 recovers by sending

$$S_{P_1}(\{PCS_{P_2}((m,4),P_1,T), PCS_{P_3}((m,4),P_1,T), \\ PCS_{P_4}((m,4),P_1,T)\}, S_{P_1}((m,1)))$$

and succeeds in overturning the abort decision.

B. A peculiar attack against the GM protocol

We here briefly sketch one of the "peculiar" attacks against the GM protocol in the case we have two dishonest participants out of four. The particularity of the attack is that the dishonest participants do not gain any advantage. They bring the protocol into a state where one of the two honest parties ends up with the signatures while all the other participants are unable to get all signatures. Moreover, MOCHA does not find any attack for this threshold where one of the dishonest parties gains the advantage.

We illustrate the attack where P_1 and P_4 are honest and P_1 will be flawed.

1. At the beginning P_3 aborts, but continues the protocol. We have that $S_m = \{3\}$ and $F_m = \emptyset$.

2. As soon as possible, dishonest P_2 recovers by sending

$$S_{P_2}(\{PCS_{P_1}((m,2),P_2,T), PCS_{P_3}((m,1),P_2,T), \\ PCS_{P_4}((m,1),P_2,T)\}, S_{P_2}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{3,2\}$ and $F_m = \{1\}$.

3. Honest P_1 is forced to recover by sending

$$S_{P_1}(\{PCS_{P_2}((m,3),P_1,T), PCS_{P_3}((m,3),P_1,T), \\ PCS_{P_4}((m,1),P_1,T)\}, S_{P_1}((m,1)))$$

but receives an abort reply from T. We have that $S_m = \{3,2,1\}$ and $F_m = \{1\}$.

4. Honest P_4 is forced to recover by sending

$$S_{P_4}(\{PCS_{P_1}((m,3),P_4,T),PCS_{P_2}((m,3),P_4,T),\\ PCS_{P_3}((m,3),P_4,T)\},S_{P_4}((m,1)))$$

and overturns the abort decision.