Optimistic Asynchronous Multi-Party Contract Signing with Reduced Number of Rounds

Birgit Baum-Waidner

Entrust Technologies (Switzerland)
bbaum@ieee.org, birgit.baum@entrust.com

May 27th, 2001

Abstract. Optimistic asynchronous multi-party contract signing protocols have received attention in recent years as a compromise between efficient protocols and protocols avoiding a third party as a bottleneck of security. "Optimistic" roughly means: in case all participants are honest and receive the messages from the other participants as expected, the third party is not involved at all. The best solutions known so far terminate within $t + 2$ rounds in the optimistic case, for any fixed set of n signatories and allowing up to $t < n$ dishonest signatories. The protocols presented here achieve a major improvement compared to the state of the art: The number of rounds R is reduced from $O(t)$ to $O(1)$ for all $n \geq 2t + 1$, and for $n < 2t + 1$, R grows remarkably slowly compared with numbers of rounds in $O(t)$: If $t \approx \frac{k}{k+1} n$ then $R \approx 2k$.[1]

1 Introduction

A *contract signing protocol* is a protocol that allows n signatories to sign a contract text such that, even if up to t, $t < n$, of them are dishonest, either *all* honest signatories obtain a signed contract, or nobody obtains it [3, 9].[2] Dishonest signatories can arbitrarily deviate from their protocol (Byzantine model). We assume an *asynchronous* network, i.e., there are no upper bounds on network delays.

Multi-party contract signing has obvious applications in secure electronic commerce, and is the basis for the solution of many related fairness problems, like multi-party certified mail [2].

By using a *third party,* T, that is always honest, the problem can be trivially solved [19]: T collects a digital signature from each of the n signatories, and either redistributes the n signatures, or aborts the contract in case not all n signatures arrive. Security depends fully on T; therefore most research has been focused on getting rid of T as trust and performance bottleneck.

Unfortunately, one cannot get rid of T completely: For $n = 2$ no deterministic protocol without third party exists [12], and each probabilistic protocol has an error probability at least linear in the number of rounds [8]. Therefore the goal is to minimize the involvement of T as much as possible: *Optimistic* protocols depend on a third party

[1] A short version without proofs will be published in [1].
[2] A precise definition of the asynchronous n-party contract signing problem is given in Sect. 3.

T, but in such a way that T is *not* actively involved in case all signatories are honest and "patient enough;" only for recovery purposes T might become active [4, 8, 17].

Optimistic contract signing protocols have been first described for *synchronous* networks [3, 4, 17] which is the stronger and more unrealistic model but allows protocols to manage within 2 normal phases plus 2 recovery phases, for any number of signatories and any fault assumption. 2-party protocols for *asynchronous* networks have been described in [5, 13, 18]. The n-party case was first investigated in [6]. Independently, but subsequently, protocols for the 3-party and n-party case were proposed in [13] and [14], respectively, requiring $O(n^2)$ rounds for $t = n - 1$. In [7], an optimistic asynchronous n-party contract signing protocol was presented requiring $t + 2$ rounds only, and $t + 4$ in the worst case, which is the best result achieved so far and round-optimal in case of $n = t + 1$ (as shown in [14]).

The protocols presented here achieve a significant improvement in case $t < n - 1$: The number of rounds, R, in the optimistic case is

$$R = \begin{cases} 2, & \text{if } n \geq 2t + 1; \\ 2\lfloor \frac{t+1}{n-t} \rfloor + \min(2, (t+1) \mod (n-t)) & \text{if } t + 2 < n \leq 2t; \\ t + 1, & \text{if } n = t + 2; \\ t + 2, & \text{if } n = t + 1; \end{cases} \quad (1)$$

This means a major improvement compared to the state of the art: R is reduced from $O(t)$ to $O(1)$ for all $n \geq 2t + 1$, and for $n \leq 2t$, R grows remarkably slowly compared with numbers of rounds in $O(t)$: If $t \approx \frac{k}{k+1} n$ then $R \approx 2k$.

This enables higher numbers of participants for practical use.

2 Model and Notation

Our model and notation are identical to those in [7]:

Let $P_1, \ldots, P_n$ denote the *signatories,* T a *third party,* and $V_1, \ldots, V_{n'}$ the potential *verifiers* of signed contracts. Each *party* represents a machine that executes a certain protocol and that serves a specific user (e.g., human being or another protocol). Parties can receive local *inputs* from their users (e.g., the command to sign a certain contract) and can generate certain local *outputs* for their users (the result of the protocol, e.g. a message that the contract was signed successfully).

The signatories P_i and third party T are able to *digitally sign* messages, and all parties are able to verify their signatures [15]. The signature on message m associated with P_X (or T) is denoted by $\mathsf{sign}_X(m)$ (or $\mathsf{sign}_T(m)$). In our complexity analyses we assume that $\mathsf{sign}(m)$ has constant length, independent of the length of m (e.g., because m is hashed before signing [16]). We abstract from the error probabilities introduced by cryptographic signature schemes and assume that signatures are unforgeable (the translation into a cryptographic model is straightforward).

Adversary model. We assume that up to t (for a given t, $0 < t < n$) of the n signatories, and for some requirements also T and all verifiers might be *dishonest.* Dishonest parties can behave arbitrarily and be coordinated by a single party, called the *adversary.* Security properties must always hold for *all* adversaries.

Network. All messages sent to or from T or any V_i are reliably delivered, eventually. Messages between any P_i and P_j might never be delivered. We do not require any particular level of synchronization, nor do we assume that messages are delivered in order. The decision on which of all sent messages to deliver next is taken by the adversary. The adversary can read all messages from, and can insert additional messages into, all channels.

Rounds. In contrast to the "phases" in synchronous networks, the rounds in asynchronous networks are not synchronized. Each message is sent and received in the context of a certain round and will usually be based on messages of previous rounds, or on the local decision no longer to wait for any messages.

All-honest case. This is a special case where we assume that *all* n signatories are honest and *all* messages sent are reliably delivered, eventually.

We make use of a few conventions, in order to simplify presentation:

Local timeouts. If we say that *"X waits for certain messages, but can stop waiting any time"* we mean more precisely the following: X accepts a special input wakeup from its user, for each protocol execution. "Waiting for certain messages" means that X continues to receive messages but does not proceed in the protocol run until either the messages have been received as expected, or wakeup is input.

3 Definitions

Definition 1 (Asynchronous Multi-party Contract Signing). [3] An *Asynchronous Multi-Party Contract Signing Scheme* (asynchronous MPCS) consists of two protocols:

- $\mathsf{sign}[P_1, \ldots, P_n]$, for signing a contract with signatories $P_1, \ldots, P_n$. $\mathsf{sign}[]$ might involve an additional party, T, in which case we call it an *MPCS with third party.*
- $\mathsf{verify}[P_i, V_j]$, to allow P_i to show its contract to any verifier, V_j, $j \in \{1, \ldots, n'\}$. $\mathsf{verify}[]$ never involves T, i.e., it is always a 2-party protocol only.

A signatory P_i starts $\mathsf{sign}[]$ on its local input $(decision_i, tid_i, terms_i, contr_i)$, $decision_i \in \{\mathsf{sign}, \mathsf{reject}\}$.

tid_i is a transaction identifier which should be unique for each execution of $\mathsf{sign}[]$. $terms_i$ contains information about the protocol used including the number and the identities of the participants and of T, their public keys, and the assumed maximum number t of dishonest signatories. $contr_i$ is the contract text to be signed. To avoid (unintended or malicious) collisions, we require P_i to refuse concluding a contract by executing a protocol with the same tid_i and $terms_i$ as of one in which P_i is or has already been participating. If necessary, the protocol might be re-run with a new tid_i. This requirement allows protocol executions with identical tid and different $terms$ to run independently

[3] In contrast to [7], we assume that a party not willing to sign might have to be present in the protocol, as this is the case for the protocols presented in Section 4. The protocols presented in Section 6 will not need this presence.

without impacting each other. For practical reasons, this makes sense because the requirement for unique tids can be removed.[4]

P_i may receive a local input $(\mathsf{wakeup}, tid_i, terms_i)$ any time, in order to enforce progress of sign[].

Upon termination sign[] produces a local output $(tid_i, terms_i, contr_i, d_i)$ for P_i, with $d_i \in \{\mathsf{signed}, \mathsf{failed}\}$. We will simply say "$P_i$ decides d_i."

To show the contract, a signatory P_i may start verify[] with verifier V_j on P_i's local input $(\mathsf{show}, V_j, tid_i, terms_i, contr_i)$. A verifier V_j starts verify[] on a local input $(\mathsf{verify}, P_i, tid_V, terms_V, contr_V)$. V_j may receive a local input $(\mathsf{wakeup}, tid_V, terms_V)$ any time. Upon termination verify$[P_i, V_j]$ produces a local output $(tid_V, terms_V, contr_V, d_V)$ with $d_V \in \{\mathsf{signed}, \mathsf{verify\_failed}\}$ for V_j. We will simply say "V_j decides d_V." No output is produced for P_i.

Note that the local service interfaces do not show protocol information like exchanged messages or information proving a valid contract, although that information is produced by the protocols themselves and serves as the basis for the outputs. This way also exotic solutions are included, e.g., that a proof of a contract cannot necessarily be handled as one piece of information but might require a protocol itself which even might have to interfere with the contract signing protocol.

The following requirements must be satisfied:

- **(R1)** *Correct execution.* In the all-honest case, if all signatories start with the same input (sign, tid, $terms$, $contr$), and no signatory receives (wakeup, tid, $terms$), then all signatories terminate and decide signed.
- **(R2)** *Unforgeability.* If an honest P_i never received input $(\mathsf{sign}, tid, terms, contr)$ then *no* honest V_j that receives input $(\mathsf{verify}, P_{i'}, tid, terms, contr)$, for any $P_{i'}$, will decide signed. (Note that this does not assume an honest T.)
- **(R3)** *Verifiability of valid contracts.* If an honest P_i decides signed on input (sign, tid, $terms$, $contr$), and later P_i receives input $(\mathsf{show}, V_j, tid, terms, contr)$ and honest V_j receives input $(\mathsf{verify}, P_i, tid, terms, contr)$ and does not receive (wakeup, tid, $terms$) afterwards then V_j will decide signed.
- **(R4)** *No surprises with invalid contracts.* If T is honest, and an honest P_i received input $(\mathsf{sign}, tid, terms, contr)$ but decided failed then no honest verifier V_j receiving $(\mathsf{verify}, P_{i'}, tid, terms, contr)$, for any $P_{i'}$, will decide signed.
- **(R5)** *Termination of* sign[]. If T is honest then each honest P_i that receives sign and wakeup (for the same tid and $terms$) will terminate eventually.
- **(R6)** *Termination of* verify[]. Each honest V_j that receives verify and then wakeup, for the same tid and $terms$, and each honest P_i that receives show will terminate eventually.

Definition 2 (Optimistic Protocol). An MPCS with third party T is called *optimistic on agreement* if in the all-honest case, if all signatories receive input (sign, tid, $terms$, $contr$) and none receives (wakeup, tid, $terms$), the protocol terminates without T ever sending or receiving any messages. It is called *optimistic on disagreement* if in the all-honest case, if some signatories do *not* receive input (sign, tid, $terms$, $contr$), and none receives (wakeup, tid, $terms$), the protocol terminates without T ever sending or

[4] Different sets of signatories may use the same tid as here $terms$ will be different.

receiving any messages. It is called *optimistic* if it is optimistic on agreement and on disagreement. We will call messages "optimistic" if they would appear in the "optimistic case", i.e., in a protocol execution where the third party is not needed.

4 Scheme Requiring Presence of Unwilling Parties

The following Scheme 1 solves the multi-party contract signing problem, is optimistic on agreement, and terminates in a number of locally defined rounds R as defined in Eq. 1. It requires the presence and actions of honest parties even if they do not want to sign the contract. In Section 6 we will remove this disadvantage thereby requiring one additional round. The drastic improvement towards [7] in the number of rounds results from the rule that T does not always answer requests "immediately" but has to wait for further requests, if the scheme says that such would be sent in case the sender is honest.

Ignoring all details, and somewhat simplified, Scheme 1 works as follows:

In Round 1 each signatory that wants to sign the contract (has received a local input $(\mathsf{sign}, tid, terms, contr)$) starts the protocol signing its round-1-message and broadcasts it. This message will be interpreted as an "intent to sign the contract". In each subsequent round the signatory tries to collect all signatures from the previous round, countersigns this set of n signatures, and broadcasts it.[5] The result of the R-th round becomes the real contract. The messages mentioned so far build the "optimistic part".

If some signatory does not want to sign the contract (i.e. if its local input was $(\mathsf{reject}, tid, terms, contr)$), it signs a reject (rather than its round-1-message), sends it to T and stops.

A signatory that becomes tired of waiting for some signatures in the context of some round (i.e. receives the local input $(\mathsf{wakeup}, tid, terms)$) sends the relevant part of information received so far to the honest T, stops sending further messages, waits for T's answer (which will contain aborted or signed) and stops protocol execution for these tid and $terms$ after receiving the response from T. Each response from T is final for all honest signatories, even if T itself does not handle the result as final yet – though T will handle it as final e.g. if it "knows" that some honest signatory must be among the recipients of that response.

The protocol steps provided for T use the assumptions that there are at least $n - t$ honest signatories in total, and that any set of $t+1$ signatories contains at least one honest signatory, since only up to t signatories are assumed to be dishonest. If T receives a resolve or a reject from a signatory P_i in the context of some round $r < R$, T uses the fact that P_i, if honest, would not send any messages in the context of a round $r' > r$, nor would P_i send both a reject and a resolve in the same protocol execution (all such messages might be used as dishonesty-proofs by T). Since, if honest P_i contacted T, all other honest signatories not having contacted T so far would eventually contact T, too, by anyway sending reject (in the context of Round 1) or due to a missing Round r or $r + 1$ message (e.g., at the latest the Round-($r + 1$)-message from honest P_i which cannot be sent any more). So it is safe for T to wait for such a number of messages.

In case P_d is dishonest and contacts T pretending missing messages without justification, it might happen that P_d will never get an answer.

[5] The real protocol does this more efficiently, in order to keep the messages short.

If r is the highest round number in the context of which T received messages, T will consider possibly honest (and will respond to) at most those having contacted T for Rounds $r-1$ and r.

If T receives a request, it either answers immediately or waits for further messages as long as it is sure that such will still come. T must stop waiting as soon as it considers possible that all signatories which still might send messages could as well be dishonest and might never send anything. T sends responses immediately if the result is already, or just becomes, clear and final: signed as soon as T has proofs that all which were sent an aborted, if any, are dishonest anyway and T has a Round-r-message with $r > 1$, seeing all n "intentions" to sign the contract. aborted becomes final if T has proofs that one among those which will have been sent an aborted must be honest. In all other cases, after sufficient messages, T answers the non-final aborted.

Generally, T can get requests at any time in the context of any round, without any restriction. The context to which T responds any requests is always the highest round number for which T has collected requests since the last response.

Note that there is no synchronisation of rounds. Each signatory which wants to sign the contract (i.e., having received a local input $(\mathsf{sign}, tid, terms, contr)$) handles its current round locally until it can start the next round or gets "impatient" (i.e. receives the local input $(\mathsf{wakeup}, tid, terms)$).

Among others, the protocol has the following properties.

As long as T considers a signatory possibly honest, to which it had send an aborted, T does not send signed to other signatories.

In all cases where T, at the time it has to answer in the context of Round r, considers possible that a signatory which got the answer aborted in the context of Round $r-1$ might be honest, T responds aborted. Thus, the protocol would fail if T could be forced to answer aborted in all rounds until round $R-1$ to dishonest signatories which might themselves get the contract but forced T to answer aborted to a honest signatory in the context of Round R. As the proof implies, this cannot happen because R is constructed in a way (Eq. 1) that the number of dishonest signatories can never be sufficiently high to perform this attack. If T receives any request in the context of the *last* round, T responds signed. R is sufficiently high such that this response is safe.

signed is always final for T. T can switch its local guessed result aborted to signed, but only if T can conclude that only dishonest signatories got aborted so far. Each result, as soon as received by a (honest) signatory, is final at once. Thus, if a honest signatory got aborted while this was not final yet for T, the protocol must guarantee that no signatory will ever get signed. This is the case for Scheme 1.

Scheme 1 (Scheme Requiring Presence of Unwilling Parties)

Protocol "sign" for honest P_i:[6]

The protocol is given by the following rules; P_i applies them, repeatedly, until it *stops*. Let $c_i := (tid_i, terms_i, contr_i)$.

The protocol will proceed in locally defined rounds. R is the number of rounds as defined in Eq. 1 using the parameters n and t given in $terms_i$. Let $r := 1$ a local round

[6] These rules are similar to those in [7]. The most significant difference lies in the protocol for T.

counter for P_i. It will be increased only by P_i applying its protocol rules itself as long as P_i does not get "impatient" (i.e., receives the local input $(\mathsf{wakeup}, \mathit{tid}_i, \mathit{terms}_i)$ – there are no timeouts within the protocol itself. Let $\mathit{raised\_exception} := \mathsf{false}$ a Boolean variable indicating whether P_i contacted T or not. Both are initialized before executing any rule. Any $\mathsf{sign}_i(c_i, r, \mathsf{prev\_rnd\_ok})$ will mean P_i's confirmation that it already received all optimistic messages for c_i of the Round $r-1$ if such exists. Any $M_{r,i}$ is the complete vector of such confirmations from all signatories for the same r and c_i, and any $\mathsf{sign}_i(M_{r-1,i}, r, \mathsf{vec\_ok}))$ means P_i's confirmation that it has successfully collected such a complete vector for the previous round $r-1$. Let $M_{0,i} := \mathsf{nil}$, for all i. From terms_i, P_i concludes identities and public keys of the other signatories and of T. Note that information is needed in clear only if the recipient, which has to check the signature, does not have or can deviate the original information before it was signed. This holds for recipient T only. As usual, we assume implicitly that all messages contain additional implementation relevant information about the protocol and its execution.

- **Rule S0:** If $\mathit{decision}_i = \mathsf{reject}$ then:
  - P_i sends $\mathit{reject}_i = (c_i, i, \mathsf{sign}_i(c_i, \mathsf{reject}))$ to T.
  - P_i decides failed and *stops.*
- **Rule S1:** If $\mathit{raised\_exception} = \mathsf{false}$ and $r = 1$ then:
  - P_i sends $m_{1,i} := \mathsf{sign}_i(c_i, 1, \mathsf{prev\_rnd\_ok})$ to all signatories. We call those messages *optimistic*.
  - From all received messages of type $m_{1,j}$, from those signatories, P_i tries to compile full and consistent vectors
    $M_{1,i} := (\mathsf{sign}_1(c_i, 1, \mathsf{prev\_rnd\_ok}), \ldots, \mathsf{sign}_n(c_i, 1, \mathsf{prev\_rnd\_ok}))$ and
    $X_{1,i} := M_{1,i}$. (NB: here P_i has verified that the sender agrees on c_i.)
    If this succeeds P_i sets $r := 2$.

  P_i will not wait infinitely. At any time P_i can stop waiting for any missing $m_{1,j}$ (i.e., the user of P_i might enter wakeup any time), in which case it sets $\mathit{raised\_exception} := \mathsf{true}$ and sends $\mathit{resolve}_{1,i} := (c_i, 1, i, \mathsf{sign}_i(m_{1,i}, \mathsf{resolve}))$ to T.
- **Rule S2:** If $\mathit{raised\_exception} = \mathsf{false}$ and $2 \le r \le R$ then:
  - P_i sends $m_{r,i} := (\mathsf{sign}_i(M_{r-1,i}, r, \mathsf{vec\_ok}), \mathsf{sign}_i(c_i, r, \mathsf{prev\_rnd\_ok}))$ to all signatories. We call those messages and their signed pieces *optimistic*.
  - From all received messages of type $m_{r,j}$ it tries to compile full and consistent vectors

    $M_{r,i} = (\mathsf{sign}_1(c_i, r, \mathsf{prev\_rnd\_ok}), \ldots, \mathsf{sign}_n(c_i, r, \mathsf{prev\_rnd\_ok}))$
    $X_{r,i} := (\mathsf{sign}_1(M_{r-1,i}, r, \mathsf{vec\_ok}), \ldots, \mathsf{sign}_n(M_{r-1,i}, r, \mathsf{vec\_ok}))$

    If this succeeds and if

    $r < R$ then it sets $r := r + 1$; or if
    $r = R$ then it decides signed, sets $C_i := (c_i, M_{r,i})$, and *stops.*

  P_i will not wait infinitely. At any time P_i can stop waiting for any missing $m_{r,j}$. In this case P_i sets $\mathit{raised\_exception} := \mathsf{true}$ and sends
  $\mathit{resolve}_{r,i} := (c_i, r, i, \mathsf{sign}_i(X_{r-1,i}, \mathsf{resolve}), X_{r-1,i}, M_{r-2,i})$ to T.

- **Rule S3:** If $raised\_exception$ = true then:
  P_i waits for a message from T (without sending any messages itself).
  This can be any of $signed_{r',j} = (resolve_{r',j}, \mathsf{sign}_T(c, r', j, \mathsf{signed}))$
  or $aborted_{r',j} = (c, r', j, \mathsf{sign}_T(c, r', j, \mathsf{aborted}))$.
  On receiving one, P_i decides signed in the former or failed in the latter case. In case it decides signed it sets $C_i := signed_{r',j}$. In both cases, it *stops* now.

Protocol "sign" for third party T:

We assume T receives a message $resolve_{r,i}$ or $reject_i$. Let $message_{r,i}$ be that message. Let c_i be the triple $(tid,\ terms,\ contr_i)$ contained in clear in $message_{r,i}$. Let R be as defined in Eq. 1, according to the parameters n and t as specified in $terms$.

If this is the first time T receives a message in the context of this tid and $terms$, T initializes Boolean variables $Signed$:= false, $IsFinalResult$:= false, a String variable $TheResult$:= null, an Integer variable $CurrentContextRound$:= 0, and sets $ReceivedQuests := \emptyset$, $RecogDishonestContact := \emptyset$, $MaybeHonestRejected := \emptyset$, $MaybeHonestGotAborted := \emptyset$, and $MaybeHonestUnanswered := \emptyset$.

$Signed$ indicates T's currently guessed result, signed (by $Signed$ = true) or aborted (by $Signed$ = false). $IsFinalResult$ tells if $Signed$ is final or still might change. (Remark: A $Signed$ = true will always be final, and a $Signed$ = false will not change unless T has proofs that no honest signatory got any $aborted_{r',j}$.)

$TheResult$ is the final result in case of $IsFinalResult$ = true. $CurrentContextRound$ gives the highest number of rounds in the context of which T has already received a message. $RecogDishonestContact$ is the set of signatories which have contacted T and were identified by T as dishonest in the meanwhile. $ReceivedQuests$ collects the accepted messages. $MaybeHonestRejected$ is the set of signatories which sent reject to T in Round 1 and do not belong to $RecogDishonestContact$. $MaybeHonestGotAborted$ is the set of signatories which earlier got from T the response aborted but do not belong to $RecogDishonestContact$. $MaybeHonestUnanswered$ is the set of signatories currently waiting for an answer from T not belonging[7] to $RecogDishonestContact$. All those sets of signatories are handled in a way that they will never overlap.

The following sequential steps process a $message_{r,i}$, and only one such message at a time. As soon as T can stop processing $message_{r,i}$ according to one of the steps, the next message can be processed. If more than one arrive, they will be processed in any order (by performing those sequential steps for each such message). $message_{r,i}$ is considered having been sent in the context of Round r, and of Round $r = 1$ in case $message_{r,i}$ = $reject_i$. Since the model is asynchronous, T might receive messages at any time for any r and not necessarily in increasing order.

- **Step T0a:** *(T accepts only one message directly from each P_i, and only from signatories not already identified as dishonest by T.)* If $[P_i \in MaybeHonestRejected \cup MaybeHonestGotAborted \cup MaybeHonestUnanswered \cup RecogDishonestContact] \vee [message_{r,i}$ does not look like a syntactically correct message built according to the rules for P_i

[7] which not necessarily means they are considered honest – T just "doesn't know."

(e.g., if P_i does not belong to the signatories specified in *terms*)]], then $message_{r,i}$ is ignored for these *tid* and *terms*, and processing that $message_{r,i}$ is *stopped.* Otherwise,

- $\mathit{ReceivedQuests} := \mathit{ReceivedQuests} \cup \{message_{r,i}\}$ and

in case of $message_{r,i}$ = $reject_i$

- $\mathit{MaybeHonestRejected} := \mathit{MaybeHonestRejected} \cup \{P_i\}$

in case of $message_{r,i}$ = $resolve_{r,i}$

- $\mathit{MaybeHonestUnanswered} := \mathit{MaybeHonestUnanswered} \cup \{P_i\}$
- $\mathit{CurrentContextRound} := \max(\mathit{CurrentContextRound}, r)$

and (in both cases) T performs

– **Step T0b:** *(T tries to conclude dishonesty for as many as possible P_j, by investigating their messages directly contained in ReceivedQuests. The sets are changed, accordingly.)* For all $P_j \in \mathit{MaybeHonestRejected} \cup \mathit{MaybeHonestUnanswered} \cup \mathit{MaybeHonestGotAborted}$, and for all their messages $message_{r',j} \in \mathit{ReceivedQuests}$ (i.e. including the new $message_{r,i}$ accepted in T0a):
If $[r' < \mathit{CurrentContextRound} - 1] \vee [message_{r',j} = reject_j$ and $\mathit{ReceivedQuests}$ contains indirectly (i.e., within any other message contained in $\mathit{ReceivedQuests}$) any optimistic piece of information $optimisticpiece_{r'',j}$, signed by $P_j] \vee [message_{r',j} = resolve_{r',j}$ and $\mathit{ReceivedQuests}$ contains indirectly any optimistic piece of information $optimisticpiece_{r'',j}$, $r'' > r'$, signed by P_j in the context of round $r''] \vee [contr$ contained in $message_{r',j}$ is different from any $contr'$ already directly or indirectly signed by P_j within another message in $\mathit{ReceivedQuests}]$[8], then

- $\mathit{RecogDishonestContact} := \mathit{RecogDishonestContact} \cup \{P_j\}$
- $\mathit{MaybeHonestRejected} := \mathit{MaybeHonestRejected} \setminus \{P_j\}$
- $\mathit{MaybeHonestGotAborted} := \mathit{MaybeHonestGotAborted} \setminus \{P_j\}$
- $\mathit{MaybeHonestUnanswered} := \mathit{MaybeHonestUnanswered} \setminus \{P_j\}$
- If $message_{r,i}$ = $reject_i$:
  T *stops* here processing $message_{r,i}$.

If $P_i \in \mathit{RecogDishonestContact}$, as a result of Step T0b, then T *stops* processing $message_{r,i}$. Otherwise $message_{r,i}$ must be a $resolve_{r,i}$, and T performs the following:

– **Step T1:** *(If the result was already final then T sends it to all signatories in $\mathit{MaybeHonestUnanswered}$.)*
If $\mathit{IsFinalResult} = \mathsf{true}$ then

- T sends to all $P_j \in \mathit{MaybeHonestUnanswered}$ the final result $\mathit{TheResult}$
- If $\mathit{Signed} = \mathsf{false}$:
  $\mathit{MaybeHonestGotAborted} :=$
  $\mathit{MaybeHonestGotAborted} \cup \mathit{MaybeHonestUnanswered}$
- $\mathit{MaybeHonestUnanswered} := \emptyset$
- T *stops* processing $resolve_{r,i}$.

[8] These conditions are sufficient to show that P_j does not act properly according to its rules S0 . . . S3. Note that $\mathit{ReceivedQuests}$ can contain at most one message per signatory, due to T0a.

Otherwise T performs

– **Step T2a:** *(If all signatories to which T sent response* aborted *or from which T received* reject*, if any, are proven dishonest and T has proofs in $ReceivedQuests$ that all signatories had intended to sign the contract, T responds* signed *as final result.)*
If $[MaybeHonestGotAborted \cup MaybeHonestRejected = \emptyset] \wedge$ [$ReceivedQuests$ contains direct or embedded signed messages (except reject) from all n signatories to the c_i contained in $resolve_{r,i}$] then
  - $IsFinalResult := \mathsf{true}$
  - $TheResult := (resolve_{r,i}, \mathsf{sign}_T(c_i, r, i, \mathsf{signed}))$
  - $Signed := \mathsf{true}$
  - T sends to all $P_j \in MaybeHonestUnanswered$ the final result $TheResult$
  - $MaybeHonestUnanswered := \emptyset$
  - T *stops* processing $resolve_{r,i}$.

Otherwise T performs

– **Step T2b:** *(If* aborted *is a safe result due to at least one honest signatory which would have obtained it, make it final. The proof shows that this step can only be applied if $r < R$.)* If $\sharp(MaybeHonestRejected \cup MaybeHonestGotAborted \cup MaybeHonestUnanswered) \geq t + 1 - \sharp(RecogDishonestContact)$ then
  - $IsFinalResult := \mathsf{true}$ (and $Signed$ stays false)
  - $TheResult := (c_i, r, i, \mathsf{sign}_T(c_i, r, i, \mathsf{aborted}))$
  - T sends to all $P_j \in MaybeHonestUnanswered$ the final result $TheResult$
  - $MaybeHonestGotAborted :=$ $MaybeHonestGotAborted \cup MaybeHonestUnanswered$
  - $MaybeHonestUnanswered := \emptyset$
  - T *stops* Processing $resolve_{r,i}$.

Otherwise T performs

– **Step T3:** *(If T considers possible that at the same time a signatory in $MaybeHonestUnanswered$ might be honest (e.g., P_i might be honest) and all those not having sent anything to T might be dishonest (and e.g. might never send a message to T), then T must answer the current guess* aborted *though this result might not necessarily be final for T – while it will be final for any honest recipient though.)*
If $[\sharp(MaybeHonestRejected \cup MaybeHonestGotAborted$ $\cup MaybeHonestUnanswered) \geq n - t]$ then
  - T sends to all $P_j \in MaybeHonestUnanswered$ the result $(c_i, r, i, \mathsf{sign}_T(c_i, r, i, \mathsf{aborted}))$
  - $MaybeHonestGotAborted :=$ $MaybeHonestGotAborted \cup MaybeHonestUnanswered$
  - $MaybeHonestUnanswered := \emptyset$
  - T *stops* processing $resolve_{r,i}$.

Otherwise T performs

– **Step T4:** *(No condition is left allowing T to immediately respond to signatories in $MaybeHonestUnanswered$. This means those have to wait until more signatories will ask T. The proof shows that the latter will happen in case $MaybeHonestUnanswered$ actually contains honest signatories.)*

- T *stops* processing $resolve_{r,i}$.

Protocol "verify":

If P_i wants to show a signed contract on c to verifier V it sends C_i to V. V decides signed if it receives a messages C_i from P_i such that

- **Rule V1:** $C_i = (c, (\mathsf{sign}_1(c, R, \mathsf{prev\_rnd\_ok}), \ldots, \mathsf{sign}_n(c, R, \mathsf{prev\_rnd\_ok})))$, or
- **Rule V2:** $C_i = (resolve_{r,j}, \mathsf{sign}_T(c, r, j, \mathsf{signed}))$, for some $r \geq 2$, and some j.

and *stops.* On input wakeup, V outputs verify_failed and *stops.*

Theorem 1 (Security of Scheme 1). *Scheme 1 is an asynchronous MPCS with third party T for any $t < n$. It is optimistic on agreement and terminates in R rounds if T is not involved, and in $R + 2$ rounds in the worst case, with R as defined in Eq. 1.*

For the detailed proof, see Appendix C. To show that Requirements R1 - R6 are fulfilled, it makes use of the rules for the signatories of Scheme 1, and of Lemma 1.

Lemma 1. *Consider Protocol "sign" of Scheme 1: When T has, for the first time in the context of Round $R-1$, responded* aborted *for a $resolve_{R-1,i}$, then T has sent* aborted *to, or received* reject *from, or can show dishonesty-proofs for, at least $t + 1$ signatories in total.*

Lemma 1 is shown in Appendix B, fully using all properties and cases of R in Eq. 1, depending on n and t. It is especially needed to show that R is sufficiently high so that T cannot be fooled to answer aborted to a honest signatory having contacted T in the context of Round R, while another signatory would obtain the signed contract. The proof uses also the following Lemma 2.

Lemma 2. *Consider a protocol execution p such that T, for the first time in the context of Round $R - 1$, responded* aborted*, and T has not sent* aborted *to, or received* reject *from, or can show dishonesty-proofs for, $t + 1$ signatories yet. Then $R > 2$, and T responded* aborted *in the context of each round before Round $R - 1$. In the context of Round 1, T received messages (*resolve *or* reject*) from at least $n - t$ signatories. In all contexts of any two rounds r and $r + 1, 1 < r < R-2$, T received* resolve *from at least $n - t$ signatories each.*

Lemma 2 is shown in Appendix A, using the rules for T (Steps T0 - T4) and for the signatories (Rules S0 - S3). Here the construction of R depending on n and t, according to Eq. 1 can be understood as Lemma 2 gives a hint for one of the most malicious cases significant for determining a sufficiently high R (as done for Scheme 1). If it was possible that T answered aborted for contexts including Round $R - 1$, exclusively to dishonest signatories, T could be fooled to answering aborted to a honest signatory P_h having asked for Round R, while all dishonest signatories get all optimistic messages needed for their contract but just do not send one last message to P_h and pretend, towards T, missing messages in all rounds $< R$. Thus T could not distinguish this protocol execution from a crucially different protocol execution where the recipients of T's last response, sent before getting contacted by P_h, are honest and where P_h is dishonest and thus must not get the contract. Due to Lemma 1, luckily, this case cannot happen since one of them which got aborted up to context of Round $R-1$ would be honest (and thus would not send its optimistic message in Round R.)

To demonstrate that case consider $n = 13, t = 9$. According to Eq. 1, $R = 6$. We show that 5 rounds would not be sufficient: Assume $P_1, \ldots, P_9$ dishonest act towards the honest ones like in the optimistic case but do not send one last Round-R-message to them. T receives requests in the following order: For Round 1 from $P_1, \ldots, P_4$, one after the other, now T responds aborted (T3). For Round 2, a request comes from P_5, and T responds aborted immediately as the condition in T3 still holds. For Round 3, requests come from $P_6, \ldots, P_8$, and T responds aborted. T could not answer immediately because it got dishonesty-proofs for $P_1, \ldots, P_4$, reducing the left side sets in the equation for T3. For Round 4, a request comes from P_9, and T responds aborted. With 5 rounds only, the dishonest ones may have the contract, but a honest P_h asking in Round 5 would get aborted, since T has no dishonesty-proof for P_9 – it could as well have been honest, from T's point of view, thus P_h could not be sent signed. 6 rounds, however, are sufficient to prevent this (and also other) attacks.

5 Number of Messages and Rounds

Let C_s and C_b be the costs of a single and a broadcast message, respectively.

In the optimistic case, Protocol "sign" of Scheme 1 runs in R rounds, according to Eq. 1. In each round, each signatory broadcasts one message to all other signatories, resulting in costs of RnC_b. In the worst case each signatory might have one additional message exchange with T, resulting in $R+2$ rounds and costs of $RnC_b + 2nC_s$. If one assumes that each broadcast requires $n-1$ single messages we end up with costs of $(Rn(n-1)+2n)C_s = O(Rn^2)C_s$. All broadcast messages of Scheme 1 have constant length, all messages sent to or by T have length $O(n)$; thus we need $O(Rn^2)$ bits only.

According to [14] any asynchronous optimistic multi-party contract signing protocol tolerating $t = n-1$ requires a number of rounds at least linear in n. Our work shows that this lower bound actually only holds for $t = n-1$ but not for smaller t, as – for the first time – better numbers of rounds were found for this case (e.g., $R = 2$ for $n \geq 2t+1, R = 3$ for $n = 2t$).

6 Scheme Without Presence of Unwilling Parties

Scheme 1 in Section 4 requires all honest parties to be present even if they are not willing to sign the contract. This section shows how to easily modify Scheme 1 to overcome this disadvantage, thereby spending one more round in case $n - t > 1$. For $n - t = 1$, Scheme 2 is identical to that in [7], and no additional round is needed.

The absence of unwilling parties and thus of reject messages implies that T cannot rely anymore on the fact that in case a signatory P_i asking in the context of Round 1 is honest, all other honest signatories would send a reject or a resolve message, too. If honest signatories do not want the contract, they would have sent messages reject in Scheme 1 but will not send anything in Scheme 2 where messages reject do not exist anymore. Therefore T has to send a response immediately for each Round-1-message (resolve only) as long as T did not receive any requests in the context of higher rounds. As soon as the first request in the context of Round 2 or higher turns up then all parties are proven to be present since they had caused sending messages $m_{r,i}$. In this case, the same situation due to the presence of honest parties is given as in Scheme 1, thus the same rules can be applied. However, an additional round is needed for Scheme 2 for the cases $n - t > 1$ since only from Round 2 on, the same situation is given as from Round 1 on in Scheme 1.

Scheme 2 (Scheme Without Presence of Unwilling Parties) This Scheme is identical with Scheme 1 except for the following slight modifications.

– Unwilling signatories do not participate at all, they can simply be absent. Rule S0 is obsolete as no action is needed in case $decision_i$ = reject, thus no $reject_i$ will exist.
– The new number of rounds is

$$R = \begin{cases} 3, & \text{if } n \geq 2t+1; \\ 2\lfloor \frac{t+1}{n-t} \rfloor \\ + \min(2, (t+1) \mod (n-t)) +1 & \text{if } t+2 < n \leq 2t; \\ t+2, & \text{if } n \in \{t+1, t+2\}; \end{cases} \tag{2}$$

– For T: $MaybeHonestReject$ is always empty. Actions on reject and $MaybeHonestReject$ are obsolete. Step T3 gets the additional condition "$CurrentContextRound = 1 \vee$ ...", as in this case T has to answer immediately each single request.

Theorem 2 (Security of Scheme 2). *Scheme 1 is an asynchronous MPCS with third party T for any $t < n$. It is optimistic on agreement and terminates in R rounds if T is not involved, and in $R + 2$ rounds in the worst case, with R as defined in Eq. 2. It does not require presence of parties not willing to sign the contract.*

The proof for this modified protocol is analoguous to the one of Scheme 1 and therefore ommitted. It makes use of the following Lemmas, replacing Lemma 1 and Lemma 2. (The case $n - t = 1$ was anyway proven in [7].)

Lemma 3. *Consider Protocol "sign" of Scheme 2: When T has, for the first time in the context of Round $R-1$, responded* aborted *for a $resolve_{R-1,i}$, then T has sent* aborted *to, or can show dishonesty-proofs for, at least $t + 1$ signatories in total.*

Lemma 4. *Consider a protocol execution p such that T, for the first time in the context of Round $R - 1$, responded* aborted*, and T has not sent* aborted *to, or can show dishonesty-proofs for, $t + 1$ signatories yet. Then $R > 3$, and T responded* aborted *in the context of each round before Round $R-1$. In the context of Rounds 1 and 2 together, T received messages* resolve *from at least $n - t$ signatories. In all contexts of any two rounds r and $r + 1, 2 < r < R - 2$, T received* resolve *from at least $n - t$ signatories each.*

7 Variants

Scheme 2 can be transformed into an abuse-free asynchronous multi-party contract signing protocol (as defined in [13]), aided by the method presented in [7]. The result is optimistic on agreement and on disagreement and needs only 2 additional rounds and $O(n^2)$ additional messages.

8 Conclusion

Asynchronous optimistic multiparty contract signing protocols known so far required $O(t)$ rounds. In this work, protocols were found reducing the number of rounds R from $O(t)$ to $O(1)$ for all $n \geq 2t + 1$, and for $n < 2t + 1$, R grows remarkably slowly compared with numbers of rounds in $O(t)$: If $t \approx \frac{k}{k+1}n$ then $R \approx 2k$.

To enable absence of honest parties not wanting to sign the contract, at most one additional round is needed.

Current work is on a proof that R as defined in Eq. 1 is the lower bound with presence of willing parties, while Eq. 2 is the lower bound without requiring presence, for asynchronous optimistic multiparty contract signing protocols without error probability (assuming perfectly unforgeable digital signatures).

I would like to thank *Michael Waidner* for fruitful discussions.

References

1. B. Baum-Waidner: Optimistic Asynchronous Multi-Party Contract Signing with Reduced Number of Rounds; accepted for ICALP 2001, July 2001, Crete
2. N. Asokan, B. Baum-Waidner, M. Schunter, M. Waidner: Optimistic Synchronous Multi-Party Contract Signing; IBM Research Report RZ 3089 (#93135), Zürich, December 1998.
3. N. Asokan, M. Schunter, M. Waidner: Optimistic Protocols for Multi-Party Fair Exchange; IBM Research Report RZ 2892, Zürich, November 1996.
4. N. Asokan, M. Schunter, M. Waidner: Optimistic Protocols for Fair Exchange; 4th ACM Conf. on Computer and Communications Security, Zürich, April 1997, 6–17.
5. N. Asokan. V. Shoup, M. Waidner: Optimistic Fair Exchange of Digital Signatures; Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 591–606.
6. B. Baum-Waidner, M. Waidner: Asynchronous Optimistic Multi-Party Contract Signing; IBM Research Report RZ3078 (#93124), Zürich, November 1998.
7. B. Baum-Waidner, M. Waidner: Round-optimal and Abuse-free Optimistic Multi-Party Contract Signing; ICALP 2000, June 00, Geneve.
8. M. Ben-Or, O. Goldreich, S. Micali, R. L. Rivest: A Fair Protocol for Signing Contracts; IEEE Transactions on Information Theory 36/1 (1990) 40–46.
9. M. Blum: Three Applications of the Oblivious Transfer; Department of Electrical Engineering and Computer Sciences, University of California at Berkley, September 18, 1981.
10. R. Cramer, V. Shoup: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack; Crypto '98, LNCS 1462, Springer-Verlag, Berlin 1998, 13–25.
11. D. Dolev, C. Dwork, M. Naor: Non-Malleable Cryptography; 23rd Symposium on Theory of Computing (STOC) 1991, ACM, New York 1991, 542–552.
12. S. Even, Y. Yacobi: Relations Among Public Key Signature Systems; Technical Report Nr. 175, Computer Science Department, Technion, Haifa, Israel, 1980.

13. J. Garay, M. Jakobsson, P. MacKenzie: Abuse-free Optimistic Contract Signing; Crypto '99, LNCS 1666, Springer-Verlag, Berlin 1999, 449–466.
14. J. Garay, P. MacKenzie: Abuse-free Multi-party Contract Signing; Intern. Symp. on Distr. Comput. (DISC '99), LNCS 1693, Springer-Verlag, Berlin 1999, 151–165.
15. S. Goldwasser, S. Micali, R. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281–308.
16. A. Menezes, P. van Oorschot, S. Vanstone: Handbook of Applied Cryptography; CRC Press, Boca Raton 1997.
17. S. Micali: Certified E-Mail with Invisible Post Offices; 1997 RSA Conference.
18. B. Pfitzmann, M. Schunter, M. Waidner: Optimal Efficiency of Optimistic Contract Signing; ACM PODC '98, Puerto Vallarta 1998, 113–122.
19. M. Rabin: Transaction Protection by Beacons; Journal of Computer and System Sciences 27/ (1983) 256–267.

A Proof of Lemma 2

Proof. We consider p as assumed for this lemma. We know that no signed can be sent by T before aborted is sent in the context of Round $R-1$ because each signed is final (T1, T2a) and would prevent T to send the assumed aborted later.

Assume $R = 2$. According to Eq. 1, and as $t > 0$, it follows $n \geq 2t + 1$ (the case $R = t + 1$ for $t = 1, n = 3$ is included here). When T responded aborted, for the first time in Round $R - 1 = 1$, as it does in p, T must have applied T2b or T3. If T2b is applied, the condition of T2b was satisfied (with empty $MaybeHonestGotAborted$) and, because the sets are disjunct, T has sent aborted to, or received reject from, or can show dishonesty-proofs for, at least $t+1$ signatories in total, contradicting the condition in Lemma 2. If T3 is applied instead, T has sent aborted to, or received reject from, $n - t \geq t + 1$ signatories which contradicts the same condition, too. So we have shown that $R > 2$ for p.

According to Eq. 1, and as $t > 0$, from $R > 2$ follows $n \leq 2t$ (the cases $R = t + 1$ and $R = t + 2$ for which $R > 2$ are included here, i.e. all cases except $t = 1, n = 3$). Consider again Round 1. First, we show that T must respond aborted in the context of Round 1 at least once, in p, before T is contacted for a context of a Round $s > 1$. Otherwise, whenever T would first receive a $resolve$ in the context of such a Round $s > 1$ (and we know there is at least one in the context of Round $R - 1$, in p) before receiving $resolve$ from enough signatories in the context of Round 1 to make T answer, or if T had got nothing at all or only reject in the context of Round 1 so far, then such a Round-(s)-message, as it contains optimistic Round-($s - 1$)-messages from all signatories, shows that at least all those which have sent reject in the context of Round 1 must be dishonest, as they must not have sent more than reject (S0, and T learns this in T0b), and the remaining (if any) in $MaybeHonestUnanswered$ had not got an answer yet. So, for the very first answer of T, all conditions would be fulfilled in T2a to switch the result to the final result signed. This would contradict the assumption for p that T sends aborted in the context of Round $R - 1$. So we have shown that T must have responded aborted in the context of Round 1.

Now we want to show that, in the context of Round 1, T received messages (resolve or reject) from at least $n - t$ signatories. Remember that $n \leq 2t$ and thus $n - t \leq t$. We assume that T did not receive so many of those messages before it responded aborted (see above) in the context of Round 1. At the beginning, $MaybeHonestGotAborted$ is empty, and there is no final result. T cannot have responded signed before responding any aborted in Round $R - 1 > 1$ in p, since a result signed would be final (T1, T2a). T1 and T2a cannot have been applied for the first response in the context Round 1 as the result can neither have been already final before the very first answer nor consist of signed. Since $MaybeHonestGotAborted$ was empty, we can conclude that T3 cannot have been applied, either, since otherwise $n - t$ signatories would have sent resolve or reject. Since T actually responded in the context of Round 1, T must have applied T2b then (the only rule left to respond aborted) when responding for the first time, which means that the condition for T2b must have been satisfied. However, this contradicts the assumption of p that T has not sent aborted to, or received reject from, or can show dishonesty-proofs for, $t + 1$ signatories yet. So our assumption was wrong. Thus we have shown that in the context of Round 1, T had received messages (resolve or reject) from at least $n - t$ signatories.

Next we show that T actually responds aborted in the context of each round until Round $R - 1$. Consider Rounds $r > 1$: T must receive resolve, and answer aborted, in the contexts of all such rounds before $R - 1$, strictly one round after the other, since a skipped response in the context of Round s would mean that any resolve sent in a context after Round s (and there is one, at the latest in Round $R - 1$, according to the assumed p of this lemma) would contain messages which prove that all signatories having asked T before the context of Round s sent messages after they should have stopped, and thus are dishonest, according to Rule S3. This would be learnt by T when applying T0b so that all conditions would be fulfilled to immediately applying T2a which would switch the result to the final signed. This would contradict the assumption that T sends aborted in the context of Round $R - 1$, in p. So we have shown that T sends aborted in the context of each round until $R - 1$.

Finally we want to show the last statement of this lemma, i.e. that, in the context of any two Rounds r and $r + 1, 1 < r < R - 2$, T received resolve from at least $n - t$ signatories.

We consider the conditions for T for sending aborted, according to p in all rounds $r < R - 1$. T1, T2b and T3 could produce that result. $MaybeHonestRejected$ always gets empty as soon as T has been contacted in the context of any Round $r > 1$, due to T0b. There is always at least one signatory in $MaybeHonestGotAborted$ which got aborted in the context of $r - 1$, as otherwise each resolve in the context of Round r would allow to apply T2a (due to the first condition in T0b), switching the result to signed. T2b cannot have been applied since its condition would contradict the assumption of this lemma, and thus not T1, either, since only T2b could make aborted final before. This means, only T3 was applicable when aborted was answered to the signatories in $MaybeHonestUnanswered$.

At each point where T can apply T3 to send aborted, it holds for T: $\sharp(MaybeHonestRejected \cup MaybeHonestGotAborted \cup MaybeHonestUnanswered) \geq n - t$ is fulfilled. In the context of Round $r + 1$, $MaybeHonestRejected$ must be $\emptyset$, according to T0b, because all resolve in the context of $r + 1 > 2$ reveal that all signatories having sent reject have continued sending messages violating Rule S0. According to our p and T0b, $MaybeHonestUnanswered$ can only contain signatories having asked in the context of Round $r+1$ since the ones contained before have already been answered. (Remember we have shown that in our p, the resolves come strongly rounds by rounds as otherwise the result would switch to signed.) $MaybeHonestGotAborted$ must only contain signatories having asked in the context of Round $r+1$ and already got an answer before T was asked to the same round again, or having asked in the context of Round r. According to Step T0b, $MaybeHonestGotAborted$ cannot contain any signatories having asked in the context of earlier rounds than Round r. Thus, $MaybeHonestGotAborted$ and $MaybeHonestUnanswered$ can altogether only contain signatories having asked in the contexts of Rounds r or $r+1$ while $MaybeHonestRejected$ is empty. Therefore, and because T applied T3, the number of signatories having asked for rounds r and $r+1$ together must have been at least $n-t$. □

B Proof of Lemma 1

We show Lemma 1:

Proof. Assumption: there exists a protocol execution, p, so that T has received the first one of such a $\mathsf{resolve}_{R-1,i}$, from a P_i, to which it immediately had to respond aborted, but T had sent aborted to, or received reject from, or can show dishonesty-proofs for, at most t signatories only. This fulfills the assumption for the protocol execution p in Lemma 2.

From Lemma 2 follows that $R > 2$ (which means that we have already shown Lemma 1 for $R = 2$). Further it follows from Lemma 2 that T responded aborted in the context of each round before Round $R-1$, and that in the context of Round 1, T received messages (resolve or reject) from at least $n-t$ signatories, and in the contexts of any two rounds r and $r+1, 1 < r < R-2, T$ received resolve from at least $n-t$ signatories.

Because fulfilling the condition for T2b would have contradicted the assumed p, Step T2b can never have been applied, and since thus the result could not become final, T1 was not applied. Thus only T3 was applied.

Thus T only answered according to T3, which means that the following holded immediately before each answer was sent:
$\sharp(MaybeHonestRejected \cup MaybeHonestGotAborted$
$\cup MaybeHonestUnanswered) \geq n-t$.

Consider $n = t+1$. According to Eq. 1, $R = t+2$. Here, $n-t = 1$, thus T3 is fulfilled immediately for each single request received in the context of each Round $1, \ldots, R-1 = t+1$, and a response was sent before the next one arrives, according to T3 and the assumption. However, these are already $t+1$ signatories, contradiction to p.

Consider $n = t+2$. According to Eq. 1, $R = t+1$. Here, $n-t = 2$. Thus, after the first signatory's message was received by T in the context of Round 1, none of T1..T3 is fulfilled, and after the second one was received in the same context, T responds due to T3. In all succeeding rounds including $R-1$, exactly one signatory's request will be received per round, fulfilling T3 immediately each time as one of the previous round is still in $MaybeHonestGotAborted$. However this means that including round $R-2=t-1$, t signatories have contacted T but at least one askes also for Round $R-1$, resulting in $t+1$ altogether. Contradiction to p.

Consider $n>t+2, n>2t$. From Eq. 1 follows that $R=2$, contradicting p.

Consider $t+2<n\leq 2t$. According to Eq. 1,
$R=2\lfloor\frac{t+1}{n-t}\rfloor+\min(2,(t+1) \mod (n-t))$.

- Assume an odd R. This implies $\min(2,(t+1) \mod (n-t))=1$ (because the first term of R is even) and $\exists k, 1\leq k$ such that $k(n-t)+1=t+1$ which, put in $R=2\lfloor\frac{t+1}{n-t}\rfloor+\min(2,(t+1) \mod (n-t))$, results in $R=2k+1$. Now we consider the requests until Round $R-1=2k$ in protocol execution p. According to Lemma 2, at least $n-t$ signatories contact T in the context of Round 1, and due to p, at least 1 signatory asks for Round $2k$. In case $k=1$, resulting in $n=2t$, there are only those two rounds to consider, thus at least $n-t+1=t+1$ signatories contacted T, which contradicts our assumed p. In case $k>1$, we have also to consider the rounds $2,..,2k-1$. Due to Lemma 2, T gets at least $(k-1)(n-t)$ requests from signatories asking for those rounds in p, plus the ones from rounds 1 and $R-1$, thus we have $(k-1)(n-t)+1+(n-t)$ which, with $k(n-t)+1=t+1$ above, results in $t+1$ signatories. Again contradiction to p.
- Assume an even R. This implies $\min(2,(t+1) \mod (n-t))\in\{0,2\}$.
  Case $\min(2,(t+1) \mod (n-t))=0$. Then $\exists k\geq 2$ (not 1 since $n\leq 2t$) such that $k(n-t)=t+1$ which results in $R=2k$. Now we consider the requests until Round $R-1=2k-1$ in protocol execution p. According to Lemma 2, at least $n-t$ signatories contact T in the context of Round 1. We have also to consider the rounds $2,..,2k-1$. Due to Lemma 2, T gets at least $(k-1)(n-t)$ messages from signatories in the context of those rounds in p, plus the ones from rounds 1 we have $(k-1)(n-t)+(n-t)$ which – with $k(n-t)=t+1$ above – results in $t+1$ signatories. Again contradiction to p.
  Case $\min(2,(t+1) \mod (n-t))=2$. Then $\exists k\geq 1$ such that $k(n-t)+2=t+1$ which results in $R=2k+2$, knowing $n-t>2$ from earlier. (A $k=0$ would have contradicted to $t+2<n\leq 2t$.) Now we consider the requests until Round $R-1=2k+1$ in protocol execution p. According to Lemma 2, T gets at least $n-t$ messages in the context of Round 1. Additionally, we have also to consider the rounds $2,..,2k+1$. Due to Lemma 2, T gets least $k(n-t)$ messages from signatories asking in the context of those rounds in p. If we add the ones from Rounds 1 we have $k(n-t)+(n-t)=k(n-t)+2+(n-t)-2$ which with $k(n-t)+2=t+1$ result in $n-1>t+1$ signatories, due to $n>t+2$. Contradiction to p. ◻

C Proof of Schema 1

Proof. Correct execution, optimistic on agreement are obviously satisfied as here all signatories apply only Rules S1 and S2 with $raised\_exception$ = false, i.e., only the optimistic messages are sent.

Termination of verify is satisfied as we have assumed that messages between P_i and V_j are delivered eventually.

Unforgeability. Assume the contrary of R2: Honest P_i never received input (sign, $tid, terms, contr$), but an honest V_j receives input ($\mathsf{verify}, P_{i'}, tid, terms, contr$), for any $P_{i'}$, and will decide signed. However, all variants of a valid contract would contain some optimistic pieces signed by all signatories, including P_i, and these signatures exist only if all signatories started the protocol with the contract's parameters $tid, terms$, $contr$, and each at least sent a message which does not contain a reject – including P_i. Contradiction to the assumption that honest P_i never receives input ($\mathsf{sign}, tid, terms$, $contr$).

Verifiability of valid contracts. The definitions of C_i in the signing protocol for P_i satisfy the conditions checked by V, and we have assumed that messages between P_i and V_j are delivered eventually.

No surprises with invalid contracts.

Assume an honest T, and an honest P_i received input ($\mathsf{sign}, tid, terms, contr$) but decided failed, and an honest verifier V receiving ($\mathsf{verify}, P_{i'}, tid, terms, contr$), for any $P_{i'}$, decides signed.

- Assume V decided signed because of Rule V1. This implies that P_i had sent $\mathsf{sign}_i(c, R, \mathsf{prev\_rnd\_ok})$ as part of $m_{R,i}$. Thus honest P_i cannot have asked T earlier than in Round R (since Rule S3 does not allow continuing sending and receiving messages from other signatories after sending a $resolve$). If T did not get any $resolve$ in the context of Round $R-1$ before receiving $resolve_{R,i}$, P_i would have received result signed (Step T0b and T2a), which is however not the case. So there must exist a first signatory P_j from which T received a $resolve_{R-1,j}$ which was answered aborted, eventually. Due to Lemma 1, as soon as T has answered P_j it holds that T has sent thus aborted to, or received reject from, or can show dishonesty-proofs for, at least $t+1$ signatories in total until that time. Since there can exist at most t dishonest signatories, there must be a honest P_h among them having sent $resolve$ or reject. If this was sent before Round R, this would be a contradiction to a decision through Rule V1 since honest P_h's optimistic Round-R- message $\mathsf{sign}_h(c, R, \mathsf{prev\_rnd\_ok})$, needed for V1, could not have been sent then (Rule S3 for honest P_h). This means T received that message from P_h in the context of Round R (it cannot be a reject then, and P_h could be, but not necessarily is, identical with P_i), before T has sent the answer aborted to P_j. In that case, however, would $resolve_{R,h}$ have made T to move all signatories, if any, from $MaybeHonestGotAborted$ and $MaybeHonestRejected$ to $RecogDishonestContact$, according to the first condition in Step T0b. Both sets had at most contained signatories having contacted T in the context of Round $R-2$ (because all others, if any, had been moved to $RecogDishonestContact$ in the meanwhile due to T0b), and would thus be empty now. Since T would not have answered yet in the context of Round $R-1$ so far, the condition for T2a would be fulfilled so that the answer to all in $MaybeHonestUnanswered$, including P_j, would be signed. Contradiction to P_j getting aborted.
- Assume V decides because of Rule V2, seeing a message $resolve_{r,j}$ and result signed. Since all results signed are final in all steps of T, it must be a result ($resolve_{r,j}, \mathsf{sign}_T(c, r, j, \mathsf{signed})$) in $TheResult$ which had been sent, when it was sent for the first time, to some P_j (and to others, if $MaybeHonestGotAborted$ con-

tained more than just P_j), applying Step T2a. (Step T1 cannot have been applied for this first answer signed). Furthermore it must have been sent then only after a non-final aborted was sent to P_i, since otherwise P_i would have got signed.
Further, we know that $resolve_{r,j}$ must contain an optimistic piece of Round-$(r-1)$ message from P_i, if there exists such a round. Assume, T received $resolve_{1,j}$, then the second condition for T2a could only be fulfilled if T received the requests from all signatories. However, T does not wait for all requests but answers aborted earlier, after sufficient requests to fulfil the conditions for T2b or T3, i.e. after requests from $min(n-t, t+1)$ signatories. Contradiction to P_j getting signed. Thus T had received $resolve_{r,j}, r > 1$ which, thus, must contain an optimistic piece of Round-$(r-1)$ message from P_i.
This means P_i can have asked T only in the context of Rounds $r-1$ or r, and T must have received and answered P_i's resolve before T answered P_j's resolve (as otherwise also P_i would have got the final signed), and T answered to P_j according to T2a. However T2a can only apply if $MaybeHonestGotAborted = \emptyset$, but P_i is in this set because it was sent aborted earlier and there is no way in all the steps for T to remove P_i from that set except P_i acts against the rules S0, ..., S3 in a way that this could be detected in Step T0b, but P_i does not violate those rules as it is honest, i.e., sent only consistent *contr* and had stopped sending any messages after contacting T.

Termination of sign[]. For each honest signatory P_i the protocol proceeds in R rounds, and each round terminates, either because the full vector of n messages arrived that allows to enter the next round, or because T is contacted and will eventually send an answer (which results in a total number of $R+2$ rounds if T is asked in the context of Round R). However, we must show here that T will actually answer to an honest P_i: We assume T is honest and honest P_i receives sign and wakeup, i.e. contacts T after getting impatient for some Round r. If T does not answer immediately on $resolve_{r,i}$ (waiting for further messages, due to T4), then also all the steps T1,..., T3 did not have effect so far for $resolve_{r,i}$ because their conditions were not satisfied. Since P_i honest obeying S0, ..., S3, i.e., sent only consistent *contr* and stopped sending any messages after contacting T, T cannot remove P_i from $MaybeHonestUnanswered$ according to any conditions in Step T0b. Thus, P_i stays there while waiting for an answer. From all this follows: $IsFinalResult =$ false (from not applying T1), $\sharp(MaybeHonestRejected \cup MaybeHonestGotAborted \cup MaybeHonestUnanswered) < t+1-\sharp(RecogDishonestContact)$ (from not applying T2b), and $[\sharp(MaybeHonestRejected \cup MaybeHonestGotAborted \cup MaybeHonestUnanswered) < n-t$ (from not applying T3). Since P_i itself is honest and will not send any messages belonging to rounds $r', r' > r$, all the remaining honest signatories – as far as not having asked before – will have to contact T at the latest in the context of Round $r+1$, if any, making T respond to P_i, eventually.

We assume, however, that P_i will never be sent an answer by T.

Consider $r < R$, so there is such a round $r+1$. We show that there are such honest signatories: $n-t$ signatories are honest for sure ($t < n$), and less of them, namely
$\sharp(MaybeHonestRejected \cup MaybeHonestGotAborted$
$\cup MaybeHonestUnanswered) < n-t$

have sent reject or resolve so far, including P_i. Thus all the remaining honest signatories will still contact T since they never get the missing piece from P_i and will get impatient, so that the left side of the equation will increase. If it did not increase to $\geq n - t$ this would mean that there must be left more than t signatories which are not in sets on the left side, so there must also be a honest signatory among them. However, all honest ones will have asked eventually, getting into such a set. Thus T3 (or some other step even earlier) can be processed so that also P_i will eventually get an answer, contradicting our assumption.

Consider $r = R$: There are no later rounds and thus not necessarily any missing pieces from P_i at other honest signatories. We assume that T (might have received but) has not ever answered any $resolve_{R-1,k}$ before the $resolve_{R,i}$ arrives. Getting $resolve_{R,i}$ then, however, T could immediately apply T2a since all signatories in $MaybeHonestRejected$ and $MaybeHonestGotAborted$ (because they had contacted T in earlier rounds than $R - 1$) would be moved to $RecogDishonestContact$ due to T0b, and we are done. So we assume that T did answer a $resolve_{R-1,k}$ before $resolve_{R,i}$ arrived. According to Lemma 1, immediately after this answer, the condition for T2b must be fulfilled, from which also follows that it was fulfilled even before that answer (since signatories are only moved between the disjunct sets on the left side while applying T2b). So T had to apply T2b for answering P_k, making the result final, so that T1 can be applied immediately for $resolve_{R,i}$. Contradiction to the assumption.

□