

Contract Signing, Optimism, and Advantage^{*}

Rohit Chadha^a John C. Mitchell^b Andre Scedrov^c
Vitaly Shmatikov^d

^a*University of Sussex*

^b*Stanford University*

^c*University of Pennsylvania*

^d*SRI International*

Abstract

A contract signing protocol lets two parties exchange digital signatures on a pre-agreed text. *Optimistic* contract signing protocols enable the signers to do so without invoking a trusted third party. However, an adjudicating third party remains available should one or both signers seek timely resolution. We analyze optimistic contract signing protocols using a game-theoretic approach and prove a fundamental impossibility result: in any fair, optimistic, timely protocol, an optimistic player yields an advantage to the opponent. The proof relies on a careful characterization of optimistic play that postpones communication to the third party.

1 Introduction

A variety of contract signing protocols have been proposed in the literature, including gradual-release two-party protocols [6,8,15] and fixed-round protocols that rely on an adjudicating “trusted third party” [2,4,22,30,33]. In this paper, we focus on fixed-round protocols that use a trusted third party optimistically, meaning that when all goes well, the third party is not needed. The reason for designing optimistic protocols is that if a protocol is widely or frequently

^{*} At the time of writing, Mitchell, Scedrov and Chadha were partially supported by OSD/ONR CIP/SW URI “Software Quality and Infrastructure Protection for Diffuse Computing” as ONR Grant N00014-01-1-0795. Additional support for Chadha from NSF Grant CCR-0098096 and EU Global Computing project “MIKADO,” for Mitchell from NSF Grant CCR-0121403, ITR/SY “Computational Logic Tools for Research and Education,” for Scedrov from NSF Grant CCR-0098096, and for Shmatikov from ONR under Grants N00014-02-1-0109 and N00014-01-1-0837.

used by many pairs of signers, the third party may become a performance bottleneck. Depending on the context, seeking resolution through the third party may delay termination, incur financial costs, or raise privacy concerns. Obviously, the value of an optimistic protocol, as opposed to one that requires a third party signature on every transaction, lies in the frequency with which “optimistic” signers can complete the protocol without using the third party.

Some useful properties of contract signing protocols are *fairness*, which means that either both parties get a signed contract, or neither does, and *timeliness*, which generally means that each party has some recourse to avoid unbounded waiting. The reason for using a trusted third party in fixed-round protocols is a basic limitation [18,31] related to the well-known impossibility of distributed consensus in the presence of faults [21]: no fixed-length two-party protocol can be fair. Although there is a trivial protocol with a trusted third party, in which both signers always send their signatures directly to it, protocols that are fair, timely, and usefully minimize demands on the third party have proven subtle to design and verify.

This paper refines previous models, formalizes properties from the literature on fixed-round two-party contract signing protocols, and establishes relationships between them. We use the set-of-traces semantics for protocols, defining each instance of the protocol as the set of all possible execution traces, arranged in a tree. The set of traces of a protocol is derived from a multiset rewriting [12] presentation of the protocol, for concreteness, although other formalisms for characterizing protocols and their sets of traces would give similar results.

Model for optimism. One modeling innovation is an *untimed* nondeterministic setting that provides a set-of-traces semantics for optimism. Intuitively, optimistic behavior in contract signing is easily described as a temporal concept: an optimistic signer is one who waits for some period of time before contacting the trusted third party. If Alice is optimistic, and Bob chooses to continue the protocol by responding to Alice, then Alice will deliberately wait for Bob’s message rather than contact the third party. Since the value of an optimistic protocol lies in what it offers to an optimistic player, we evaluate protocols subject to the assumption that one of the players follows an optimistic strategy. As a direct way of mathematically characterizing the sequence of actions that occur in optimistic play, we allow an optimistic player to deliberately give his opponent control over whether the optimist waits for a message. In other words, an optimistic player wishes to wait for a message. We allow an optimistic player to act on this wish by entering a waiting state until the opponent’s move places the optimistic player in a non-waiting state. This gives us a direct way of defining the set of traces associated with an optimistic signer, while staying within the traditional nondeterministic, untimed setting.

Impossibility result. In evaluating protocol performance for optimistic players,

we prove that in every fair, timely protocol, an optimistic player suffers a disadvantage. The importance of this result is that optimistic protocols are only useful to the extent that signers may complete the protocol optimistically without contacting the third party. In basic terms, our theorem shows that whenever a protocol allows signers to avoid the third party, an optimistic signer gives the other signer unilateral control over the outcome at some point in the execution of the protocol.

To illustrate by example, consider an online stock trading protocol with signed contracts for each trade. Suppose the broker starts the protocol, sending her commitment to sell stock to the buyer at a specific price, and the buyer responds with his commitment. To ensure timely termination, the broker also enjoys the ability to abort the exchange by contacting the trusted third party (TTP) if the buyer has not responded. Once the buyer commits to the purchase, he cannot use the committed funds for other purposes. Even if he has the option to contact the TTP immediately, an optimistic buyer will wait for some period of time for the broker to respond, hoping to resolve the transaction amicably and avoid the extra cost or potential delay associated with contacting the TTP. This waiting period may give the broker a useful window of opportunity. Once she has the buyer's commitment, the broker can wait to see if shares are available from a selling customer at a matching or lower price. The longer the buyer is inclined to wait, the greater chance the broker has to pair trades at a profit. If the broker finds the contract unprofitable, she can abort the transaction by falsely claiming to the TTP that the buyer has not responded. This broker strategy succeeds in proportion to the time that the buyer optimistically waits for the broker to continue the protocol; this time interval, if known exactly or approximately, gives the broker a period where she can decide *unilaterally* whether to abort or complete the exchange.

Since our main result only involves one run of an arbitrary contract-signing protocol, we do not need to consider sequences of protocol runs, or interleaving of concurrent runs.

The paper is organized as follows. In section 2, we summarize our semantic framework and define the class of two-party contract signing protocols with trusted third party. In section 3, we formalize protocol properties such as fairness, optimism, and timeliness. In section 4, we formalize optimistic behavior of a participant, and show that the optimistic participant is at a disadvantage in any fair, optimistic, timely protocol. The implications of the main theorem for specific contract-signing protocols in the literature are discussed in section 5, with related work discussed in section 6. We summarize our results in section 7.

Acknowledgments. We are particularly grateful to D. Malkhi for pointing out the vulnerability of optimistic players in fair exchange. We also thank

I. Cervesato, S. Even, D. Gollmann, S. Kremer, J.F. Raskin, C. Meadows, and J. Millen for interesting and helpful discussions.

2 Model

2.1 Multiset rewriting formalism

Our protocol formalism is multiset rewriting with existential quantification, MSR [12], which can be seen as an extension of some standard models of computation, *e.g.*, multiset transformation [5] and chemical abstract machine [7]. This formalism faithfully expresses the underlying assumptions of the untimed, nondeterministic, asynchronous model. A protocol definition in MSR defines the *set of all possible execution traces* for any instance of the protocol. Any other formalism, including [1,19] and others, that leads to an equivalent set of traces would support the same results about protocols [16,13]. The synchronous model with a global clock does not seem appropriate for our investigation because fixed-round contract signing protocols in the literature [2,4,22,30,33] do not rely on a global clock.

MSR syntax involves terms, facts, and rules. To specify a protocol, first choose a vocabulary, or *first-order signature*. We assume that our vocabulary contains some basic sorts such as `publicKey` for public keys and `msg` for protocol messages. As usual, the *terms* over a signature are the well-formed expressions produced by applying functions to arguments of the correct sort. A *fact* is a first-order atomic formula over the chosen signature, without free variables. Therefore, a fact is the result of applying a predicate symbol to ground terms of the correct sort. A *state* is a finite multiset of facts.

A state transition is a *rule* written using two multisets of first-order atomic formulas and existential quantification, in the syntactic form $F_1, \dots, F_k \longrightarrow \exists x_1 \dots \exists x_j. G_1, \dots, G_n$. The meaning of this rule is that if some state S contains facts obtained by a ground substitution σ from first-order atomic formulas F_1, \dots, F_k , then one possible next state is the state S^* that is similar to S , but with facts obtained by σ from F_1, \dots, F_k removed and facts obtained by σ from G_1, \dots, G_n added, where x_1, \dots, x_j are replaced by new symbols. If there are free variables in the rule $F_1, \dots, F_k \longrightarrow \exists x_1 \dots \exists x_j. G_1, \dots, G_n$, these are treated as universally quantified throughout the rule. In an application of a rule, these variables may be replaced by any ground terms.

For example, consider state $\{P(f(a)), P(b)\}$ and rule $P(x) \longrightarrow \exists z. Q(f(x), z)$. First, we instantiate this rule to $P(f(a)) \longrightarrow \exists z. Q(f(f(a)), z)$. Applying the rule, we choose a new value c for z and replace $P(f(a))$ by $Q(f(f(a)), c)$,

obtaining the next state $\{Q(f(f(a)), c), P(b)\}$.

A set of MSR rules is called a *theory*. In an interleaving semantics of concurrency, we can commute the order of application of transition rules that affect independent parts of the system:

Proposition 1 *Let $S = S_1 \uplus S_2$ be a state such that*

(i) $S' = S'_1 \uplus S_2$ is obtained from S by the application of a transition rule t_1 using ground substitution σ_1 .

(ii) $S'' = S'_1 \uplus S'_2$ is obtained from S' by the application of a transition rule t_2 using ground substitution σ_2 .

Then S'' can also be obtained from S by the application of t_2 using σ_2 followed by the application of t_1 using σ_2 .

Proof: Follows immediately from the definition of MSR. □

2.1.1 Basic sorts

Protocol participants are identified with their public keys. We use the sort `publicKey` for participants' public keys, and let k, k', k_a, k_1, \dots to range over values of this sort. Sort `msg` is used for protocol messages, and we let m_1, m_2, \dots to range over values of this sort.

In this paper, we are concerned with two-party protocols in which the participants exchange their signatures on pre-agreed contract texts. Sort `contractText` is used for the texts. We assume that participants use a globally unique identifier of the sort `uniqueIdentifier` for each protocol instance. We use n, n', \dots to range over values of this sort. As mentioned earlier, we only need to consider a single instance of the protocol.

Finally, we assume that our vocabulary contains the sort `protocolInstance` and a function:

$$\langle -, \rightarrow, \rightarrow, \rightarrow, \rightarrow \rangle : \text{publicKey} \times \text{publicKey} \times \text{publicKey} \times \\ \text{contractText} \times \text{uniqueIdentifier} \rightarrow \text{protocolInstance}.$$

Each value of the sort `protocolInstance` identifies the two participants, the trusted third party, pre-agreed contract text and the globally unique identifier of the protocol instance. We use pd, pd', \dots to range over values of `protocolInstance`. For example, protocol instance $pd = \langle k_o, k_r, k_t, m, n \rangle$ describes the protocol instance identified as n in which participants with public keys k_o and k_r are attempting to exchange signatures on the pre-agreed text m with the help of a trusted third party whose public key is k_t .

2.1.2 Timers

In our model, timers are interpreted as *local* signals, used by participants to decide when to quit waiting for a message from the other party in the protocol. They do not refer to any global time or imply synchronicity. Timers are formalized by binary *timer predicates*, whose first argument is of the sort `publicKey` and identifies the participant who receives its signal, while the second argument is one of the following three constants of the sort `timerState`: *unset*, *set*, and *timed_out*. We use ts, ts', \dots to range over constants of the sort `timerState`, and Z, Z_1, Z_2, \dots to range over timer predicates. For example, the fact $Z(k, \text{unset})$ indicates that a timer Z belonging to the participant identified with public key k in state *unset*.

2.2 Formal model of cryptography

Contract signing protocols usually employ cryptographic primitives such as encryption, hash functions and more specialized constructs such as designated-recipient signatures [22]. In general, the purpose of cryptography is to provide messages that are meaningful to some parties, but not subject to arbitrary (non-polynomial-time) computation by others. For example, encryption provides messages that are meaningful to any recipient with the decryption key, but not subject to decryption by any agent who does not possess the decryption key. The logic-based formalism of MSR cannot capture subtle distinctions between, for example, functions computable with high probability and functions computable with low or negligible probability. Instead, we must model functions as either feasibly computable, or not feasibly computable. In the remainder of this paper, we assume some fixed theory **Possess** of rules that characterize the computationally feasible operations on messages. It is assumed implicitly that in any protocol model, the roles (see section 2.3.2) will conform to the capabilities expressed in **Possess**, since no honest agent can perform any computationally infeasible action, although we do not rely on this assumption in any proof.

For each cryptographic operation used in a protocol, we assume that **Possess** contains some MSR characterization of its computability properties. To give a concrete framework for presenting these rules, let us assume some set of predicates $\mathcal{H}as = \{has^\alpha | \alpha \text{ is any sort}\}$. Since the sort α is determined by the sort of the arguments to has^α , we will not write the sort when it is either irrelevant, or clear from context. Intuitively, a rule of the form

$$has(s_1), \dots, has(s_m), F_1, \dots, F_j \longrightarrow has(t_1), \dots, has(t_n), F_1, \dots, F_j$$

means that if an agent possesses data s_1, \dots, s_m , then under conditions spec-

ified by facts F_1, \dots, F_j , it is computationally feasible for him to also learn t_1, \dots, t_n . For example, we shall always assume that if an agent possesses x , then it can make as many copies as it desires. This can be expressed by the following rule:

$$has(x) \longrightarrow has(x), has(x)$$

The familiar ‘‘Dolev-Yao’’ [17,32] rules given in [12] can be expressed as:

$$has(x), has(k) \longrightarrow has(encrypt(k, x))$$

$$has(encrypt(k, x)), has(k^{-1}), \text{Keypair}(k, k^{-1}) \longrightarrow has(x)$$

Intuitively, these rules say that if an agent possesses a message and an encryption key, it is computationally feasible for the agent to possess the encryption of the message with the key. Conversely, if an agent possesses an encrypted message and the decryption key, then it is computationally feasible for the agent to possess the plaintext. Similarly, we model invertible operations such as pairing by MSR possession rules stating that a pair may be computed from its parts and, conversely, given a pair, its parts may be computed.

As a disclaimer, we emphasize that the results in this paper are accurate statements about a protocol using cryptographic primitives only to the extent that **Possess** accurately characterizes the computationally feasible operations. In particular, protocols that distinguish between low-order polynomial computation and high-order polynomial computation, or rely on probabilistic operations in some essential way, may fall outside the scope of our analysis and may conceivably violate some of our results.

2.3 Protocol model

A protocol P is a *contract signing protocol* if it involves three parties, O (originator), R (responder), and T (trusted third party), and enables O (respectively, R) to obtain R 's signature (respectively, O 's signature) on some pre-agreed text. For brevity, we will say *signature* as a shorthand for ‘‘signature on the pre-agreed text,’’ use terms *contract signing* and *signature exchange* interchangeably, and refer to O and R as *signers*. We assume that a contract signing protocol cannot reach a state where each party (O or R) has the other's signature unless both parties (O and R) take some action. In particular, neither O nor R may obtain a binding contract without the other participating in the protocol by executing at least one protocol step.

We specify the protocol by a MSR theory. Any sequence of rules consistent

with the theory corresponds to a valid execution trace of a protocol instance. If execution traces are naturally arranged in trees, then the MSR theory defines the set of all possible execution traces as a forest of trees. To obtain the impossibility result, we choose *any* contract signing protocol P and fix it. We assume that the contract text for each instance contains a unique identifier, and consider only a run instance of P . Since only one instance is needed to obtain our impossibility result, there is no need to consider repeated or parallel runs of the protocol.

2.3.1 Communication

Following the standard assumption that the adversary controls the network and records all messages, we model communication between O and R by a unary *network predicate* N whose argument is of the sort `msg`. Once a fact $N(m)$ for some m is added to the state, it is never removed. As in contract signing protocols in the literature [4,22], we assume that channels between signers and T are inaccessible to the adversary and separate from the network between O and R (by contrast, [26] considers security of contract signing protocols under relaxed assumptions about channel security). Channels between signers and T are modeled by ternary *TTPchannel predicates*, whose arguments are of the sort `publicKey`, `publicKey` and `msg`, respectively. For example, $tc(k_o, k_t, m)$ models the channel between O and T carrying message m .

2.3.2 Role theories

A role theory specifies one of the protocol roles such as O , R or T by giving a finite list of *role state predicates* that define the internal states of the participant playing that role and the rules for advancing from state to state. Role theory also contains another, disjoint list of *timer predicates* describing the rules for the participant's timers. A participant may advance his state by "looking" at the state of his timers or the network (*i.e.*, a timer or a network predicate appears on the left side of the rule). He may also set his timer by changing the timer's state from *unset* to *set*, but he may not change it to *timed_out*.

Definition 2 *Theory A is a role theory for participant A with public key k_a , where k_a is a constant of the sort `publicKey`, if it satisfies the following:*

(i) *A includes a finite list of predicates A_0, \dots, A_n , called role state predicates, and a finite list of timer predicates, called timers of A. The two lists are disjoint.*

(ii) *A_0 is a binary predicate whose arguments are of the sort `publicKey` and `protocollInstance`, respectively. We call A_0 the initial role state predicate.*

(iii) For each rule $l \rightarrow r$ in \mathbf{A} ,

- (1) There is exactly one occurrence of a role state predicate in l , say A_i , and exactly one occurrence of a role state predicate in r , say A_j . Furthermore, $i < j$. If A_0 occurs in l , then $A_0(k_a, p) \in l$ for some term p of the sort **protocollInstance**.
- (2) If A_i is a k -ary role state predicate occurring in l , and A_j is an m -ary role state predicate occurring in r , then $m > k$. Furthermore, if $A_i(u_1, \dots, u_k) \in l$ and $A_j(v_1, \dots, v_m) \in r$, then u_q and v_q are the same terms for all $1 \leq q \leq k$.
- (3) Let $A_i(u_1, \dots, u_k) \in l$, $A_j(v_1, \dots, v_m) \in r$. Let \mathcal{MSG} be the set of terms u such that $N(u)$ or $tc(k_1, k_2, u) \in l$ for some **TTPchannel** predicate tc . For each q , v_q is derivable from u_1, \dots, u_k and \mathcal{MSG} using the rules in **Possess** (see section 2.2). Note that by the previous clause, u_q and v_q are the same terms for all $1 \leq q \leq k$.
- (4) For each timer Z of A ,
 - (a) l and r each contain at most one occurrence of Z . Occurrences are of the form $Z(k_a, ts)$, where ts is a constant of the sort **timerState**. If Z occurs in r , then it occurs in l .
 - (b) If $Z(k_a, unset) \in l$, then either $Z(k_a, unset) \in r$, or $Z(k_a, set) \in r$.
 - (c) If $Z(k_a, set) \in l$, then $Z(k_a, set) \in r$.
 - (d) If $Z(k_a, timed_out) \in l$, then $Z(k_a, timed_out) \in r$.
- (5) If $N(u) \in l$, where N is a network predicate and u is term of the sort **msg**, then $N(u) \in r$. If $tc(k_1, k_2, u) \in l$, where tc is a **TTPchannel** predicate, and terms k_1, k_2, u are of the sort **publicKey**, **publicKey**, **msg**, respectively, then $tc(k_1, k_2, u) \in r$.
- (6) For any predicate \mathcal{P} other than a role state, timer, network, or **TTPchannel** predicate, atomic formula $\mathcal{P}(t_1, \dots, t_n)$ has the same occurrences in l as in r .

Definition 3 If Z is a timer of the participant with public key k_a , then $Z(k_a, set) \rightarrow Z(k_a, timed_out)$ is the timeout rule of Z .

2.3.3 Protocol theory

Informally, a protocol theory \mathbf{P} for a given protocol is the disjoint union of six theories: \mathbf{O} , \mathbf{R} , \mathbf{T}_0 , $\mathbf{O}_{\text{timeouts}}$, $\mathbf{R}_{\text{timeouts}}$, and $\mathbf{T}_{\text{timeouts}}$, where \mathbf{O} , \mathbf{R} , \mathbf{T}_0 are role theories, and $\mathbf{O}_{\text{timeouts}}$, $\mathbf{R}_{\text{timeouts}}$, and $\mathbf{T}_{\text{timeouts}}$ are the sets of *timeout rules* for all timers of O , R , and T , respectively. For simplicity, we will combine the role theory and the timeouts of T , and call it $\mathbf{T} = \mathbf{T}_0 \cup \mathbf{T}_{\text{timeouts}}$.

Definition 4 Theory \mathbf{P} is a protocol theory for signers O and R and trusted third party T with public keys k_o, k_r, k_t , respectively, where k_o, k_r, k_t are constants of the sort **publicKey**, if $\mathbf{P} = \mathbf{O} \uplus \mathbf{R} \uplus \mathbf{T}_0 \uplus \mathbf{O}_{\text{timeouts}} \uplus \mathbf{R}_{\text{timeouts}} \uplus$

$\mathbf{T}_{\text{timeouts}}$, where

- (1) $\mathbf{O}, \mathbf{R}, \mathbf{T}_0$ are role theories for, respectively, O, R, T with respective public keys k_o, k_r, k_t .
- (2) At most one *TTPchannel* predicate, say tc , occurs in \mathbf{O} . Each occurrence of tc is of the form $tc(k_o, k_t, m)$, where m is of the sort *msg*, and $tc(k_o, k_t, m)$ cannot occur in \mathbf{R} .
- (3) At most one *TTPchannel* predicate, say tc , occurs in \mathbf{R} . Each occurrence of tc is of the form $tc(k_r, k_t, m)$, where m is of the sort *msg*, and $tc(k_r, k_t, m)$ cannot occur in \mathbf{O} .
- (4) If some *TTPchannel* predicate occurs in \mathbf{T}_0 , then it also occurs in \mathbf{O} or \mathbf{R} .
- (5) The role state predicates and the timers of O (respectively, R) do not occur in \mathbf{R} (respectively, \mathbf{O}) and \mathbf{T}_0 . The role state predicates and the timers of T do not occur in \mathbf{O} or \mathbf{R} .
- (6) $\mathbf{O}_{\text{timeouts}}, \mathbf{R}_{\text{timeouts}}$, and $\mathbf{T}_{\text{timeouts}}$ are the sets of timeout rules of all timers of O, R , and T , respectively.

2.3.4 Threat model

We are interested in guarantees provided by contract signing protocols when one of the signers misbehaves in certain ways. The trusted third party, T , is assumed to be honest. We will call the misbehaving signer the *adversary*. The adversary does not necessarily follow the protocol, and may ignore the state of the timers or stop prematurely. In principle, an adversary may gather messages from the network, store them, decompose them into fragments and construct new messages from the fragments. However, we shall only use the following capabilities in our model: quitting the protocol prematurely, ignoring the state of the timers and intercepting messages on the network. These abilities are formalized by theories $\mathbf{O}_{\text{threat}}$ and $\mathbf{R}_{\text{threat}}$ containing *dishonest rules* for O and R , respectively.

The proof of our impossibility result (see section 4) only requires that the adversary may quit the protocol prematurely, ignore the state of the timers, or intercept messages on the network. Since an adversary with additional capabilities only needs these actions in order to take advantage of an optimistic opponent, we thus obtain a *stronger* result than if we assumed a stronger adversary. On the other hand, if we were interested in proving correctness of a protocol against a more powerful adversary, we would need to extend the theories $\mathbf{O}_{\text{threat}}$ and $\mathbf{R}_{\text{threat}}$.

We now describe the rules of $\mathbf{O}_{\text{threat}}$ in more detail. The rules of $\mathbf{R}_{\text{threat}}$ are symmetric. Quitting, that is refusing to take further part in protocol execution, is a form of dishonest behavior. To model quitting from some role state,

say O_i , which is a k -ary predicate whose arguments are of the sort $\mathfrak{s}_1, \dots, \mathfrak{s}_k$ respectively, we introduce in our vocabulary a k -ary predicate $O_{dquit,i}$ whose arguments are of the sort $\mathfrak{s}_1, \dots, \mathfrak{s}_k$. We also introduce k variables x_1, \dots, x_k of the sort $\mathfrak{s}_1, \dots, \mathfrak{s}_k$ respectively, and add the following rule in $\mathbf{O}_{\text{threat}}$:

$$O_i(x_1, \dots, x_k) \longrightarrow O_{dquit,i}(x_1, \dots, x_k), M(k_o, x_1), \dots, M(k_o, x_k)$$

If a dishonest O has not quit the protocol, then O may disregard the state of some or all of the timers that govern the behavior of honest O . For example, suppose that the following rule is in \mathbf{O} (here Z is a timer predicate, and $j > i$):

$$\begin{aligned} O_i(\vec{u}), V_1(\vec{s}_1), \dots, V_k(\vec{s}_k), Z(k_o, ts) &\longrightarrow \\ O_j(\vec{u}), W_1(\vec{t}_1), \dots, W_l(\vec{t}_l), Z(k_o, ts') & \end{aligned}$$

Dishonest O may ignore timer Z :

$$O_i(\vec{u}), V_1(\vec{s}_1), \dots, V_k(\vec{s}_k) \longrightarrow O_j(\vec{u}), W_1(\vec{t}_1), \dots, W_l(\vec{t}_l)$$

If a dishonest O has not quit the protocol, then the dishonest O may also intercept (gather) messages from the network (N), or the channel between O and trusted third party (tc). In our model, we use binary predicates M whose arguments are of the sort `publicKey` and `msg`, respectively, to represent the additional memory of the dishonest participant.

Let x, x' be variables of the sort `msg`. If O_i is a k -ary role state predicate, whose arguments are of the sort $\mathfrak{s}_1, \dots, \mathfrak{s}_k$, then pick k variables x_1, \dots, x_k of the sort $\mathfrak{s}_1, \dots, \mathfrak{s}_k$ respectively. The rules for gathering messages are:

$$\begin{aligned} O_i(x_1, \dots, x_k), N(x) &\longrightarrow O_i(x_1, \dots, x_k), N(x), M(k_o, x) \\ O_i(x_1, \dots, x_k), tc(k_o, k_t, x) &\longrightarrow O_i(x_1, \dots, x_k), tc(k_o, k_t, x), M(k_o, x) \end{aligned}$$

In the above rules, the presence of the role state predicate O_i ensures that O will not intercept messages after it has quit the protocol.

2.3.5 Initial set of facts

In addition to the protocol theory and dishonest rules for the participants, a protocol specification also includes the *initial set of facts*, say S_0 , describing the initial state of the protocol execution. We assume that the participants

have agreed on the contract text m and globally unique protocol instance identifier n . S_0 is a set that contains:

- (1) Facts $O_0(k_o, pd), R_0(k_r, pd), T_0(k_t, pd)$ exactly once, where O_0, R_0, T_0 are the initial role states of \mathbf{O} , \mathbf{R} , and \mathbf{T} , respectively, and pd is the term $\langle k_o, k_r, k_t, m, n \rangle$. There is no other occurrence of a role state predicate in S_0 .
- (2) For each timer predicate Z of O , R , or T , there is exactly one occurrence of Z in S_0 .
- (3) For each timer predicate Z of O (respectively R, T), either $Z(k_o, unset)$ (respectively, $Z(k_r, t, unset)$), or $Z(k_o, set)$ (respectively, $Z(k_r, t, set)$), but not both.
- (4) $M(k_o, m), M(k_o, n), M(k_r, m), M(k_r, n)$.

2.4 Traces and continuation trees

A *state* is a finite multiset of facts. For example, the initial state S_0 may include facts $O_0(k_o, k_o^{-1}, k_r, p)$ and $R_0(k_r, k_r^{-1}, k_o, p)$ modeling the initial states of the originator and the responder in protocol p : each knows his own public and private keys, and the opponent's public key. A *trace* from state S is a chain of nodes, with the root labeled by S , each node labeled by a state, and each edge labeled by a triple $\langle t, \sigma, \mathbf{Q} \rangle$. Here \mathbf{Q} is one of $\{\mathbf{O}, \mathbf{R}, \mathbf{T}, \mathbf{O}_{\text{timeouts}}, \mathbf{R}_{\text{timeouts}}, \mathbf{O}_{\text{threat}}, \mathbf{R}_{\text{threat}}\}$, $t \in \mathbf{Q}$ is a state transition rule, and σ is a ground substitution. If $\langle t, \sigma, \mathbf{Q} \rangle$ labels the edge from a node labeled by S_1 to a node labeled by S_2 , it must be the case that the application of $t\sigma$ to S_1 produces S_2 . Any state labeling a node in this chain is said to be *reachable from S* . We will simply say that a state is *reachable* if it is reachable from the initial state S_0 .

An edge is a *dishonest move of O* if it is labeled by some $t \in \mathbf{O}_{\text{threat}}$. O is said to be *honest in the trace* if there are no dishonest moves of O in the trace. If S is reachable by a trace in which O is honest, then S is *reachable by honest O* . The definitions for R are symmetric.

Let the *continuation tree, ctr , at state S* be the tree of all possible traces from S . This tree serves as a game tree that represents the complete set of possible plays. We can see that ctr has finite depth, allowing us to reason by induction on the height above the leaves of the tree. The reason that ctr has finite depth is that we only consider a single run of a fixed-round protocol. A protocol consists of a set of roles, and each role is a finite set of multi-set rewriting rules, each rule expressing a step in the protocol. In the multi-set rewriting rule framework, each rule in a role replaces a predicate indicating the current state with a higher-numbered predicate indicating the subsequent state, preventing

any form of state looping. Further, the additional steps provided by the threat model only allow a role to move forward in the execution of a protocol, or add a fact to the set of facts known to a principal. The former action cannot lead to looping, and the latter action need only be performed once per message sent by honest parties. Thus the continuation tree from any state has finite depth.

We use subtrees of ctr to characterize the results of certain restrictions on protocol participants. Specifically, let $ctr_{[O]}$ be the tree obtained from ctr by removing all edges in $\mathbf{O} \cup \mathbf{O}_{\text{threat}}$ along with their descendants. The tree $ctr_{[O]}$ gives the set of all possible plays if O stops participating in the protocol. The definition of $ctr_{[R]}$, giving the set of all plays when R stops participating, is similar. We will say that any edge e in ctr that is labeled by a rule in \mathbf{O} or $\mathbf{O}_{\text{threat}}$ (respectively, \mathbf{R} or $\mathbf{R}_{\text{threat}}$), is under O 's control (respectively, R 's control). To model optimism of honest signers (see section 4), we will also assume that all the edges in $\mathbf{O}_{\text{timeouts}} \cup \mathbf{R}_{\text{timeouts}}$ are under control of the adversary (dishonest participant). More specifically, our model of optimism gives control over scheduling communication with the third party to the adversary. However, some possible protocols may use other timeouts that are not under control of the adversary.

3 Properties of Contract Signing Protocols

The MSR definition of the protocol determines the set of all possible execution traces, giving rise to a continuation tree. To define protocol properties such as fairness, optimism, timeliness, and advantage, we view the continuation tree as a game tree containing all possible plays, and adapt the notion of strategy from classical game theory.

For the remainder of the paper, we will assume that only one of the signers is honest. We will use A to refer to the honest signer, *i.e.*, A refers to either O , or R , depending on which of them is honest. We'll use B to refer to the other, dishonest signer.

When we mathematically characterize the degree of each player's control over the outcome of the protocol (see section 3.2.2), we will also need to consider dishonest moves when reasoning about A 's control over the protocol. The intuitive explanation is that honesty of A refers to A 's actual behavior in the protocol (what A *does* according to the protocol specification), while A 's control over the outcome refers to all potential behaviors by the signer in A 's role (*e.g.*, what A *may do* if B quits the protocol).

Following [14], we formalize strategies as truncated continuation trees. Given a set of edges E , let $ctr \setminus E$ be the tree obtained from continuation tree ctr by removing the edges in E along with their descendants. Intuitively, if E is a subset of edges of ctr under A 's control, then $ctr \setminus E$ is the set of possible plays that result if A does not use transitions in E . Similarly, we can define $ctr_{[A]} \setminus E$ (recall that $ctr_{[A]}$ is the tree of all plays if A stops participating in the protocol).

Definition 5 *Let S be a reachable state and let ctr be the continuation tree from S . Let $X \subseteq \{A, B, T\}$.*

- (1) *If E is a subset of edges of ctr such that each edge in E is under the control of some $p \in X$, then $ctr \setminus E$ is said to be a strategy for the coalition X . If there are no dishonest moves of any $p \in X$ in $ctr \setminus E$, then $ctr \setminus E$ is said to be an honest strategy.*
- (2) *If E is a subset of edges of $ctr_{[A]}$ such that each edge in E is under the control of some $p \in X$, then $ctr_{[A]} \setminus E$ is said to be an A -silent strategy for the coalition X .*

This definition corresponds to the standard game-theoretic notion of strategy. E represents the plays that the coalition X considers unfavorable, and $ctr \setminus E$ represents the continuations that X prefers. At any given state S' in $ctr \setminus E$, an edge coming out of the node labeled by S' indicates the next move for X in accordance with the strategy $ctr \setminus E$.

To define fairness and other properties, we are interested in strategies in which the coalition X drives the protocol to a state in which some property holds:

Definition 6 *If there is a strategy $ctr \setminus E$ from S for a coalition X such that all leaf nodes of $ctr \setminus E$ are labeled by states S' that satisfy some property $\phi(S')$, then X has a strategy from S to reach a state in which ϕ holds.*

The definition for A -silent strategies is similar.

Since the players' objective in the game is to obtain each other's signatures, we are interested in the states where A possesses B 's signature and the ones where B possesses A 's signature. Formally, B possesses some term u in a reachable state S if u is derivable, using the rules in **Possess**, from the terms in B 's internal role state predicate B_i in S and B 's additional memory in S given to him by the threat model. Possession is always monotonic. Moreover, possession of B increases in a transition only if B reads a message either from the network or from the channel to T . (A proof of this statement along with the proof of monotonicity has been omitted for space considerations. The

detailed proofs are available at the ftp-site ftp://ftp.cis.upenn.edu/pub/papers/scedrov/cmss_optimjlap.pdf). The definition for A is symmetric, except that the threat model does not have to be considered.

Definition 7 *If there is a strategy for coalition X such that all leaf nodes in the strategy are labeled by states in which A possesses B 's signature, then X has a strategy from S to give A B 's signature. Moreover, if $X = \{A\}$, then A is said to have a strategy to obtain B 's signature.*

3.2 Fairness, advantage, optimism, and timeliness

We now use the notion of strategy to define what it means for a contract signing protocol to be fair, optimistic, and timely, and what it means for a participant to enjoy an advantage. The definitions are quite subtle. For example, we need to draw the distinction between a *strategy* for achieving some outcome, and a *possibility* that the outcome will happen under the right circumstances. This requires introduction of a four-valued variable to characterize the degree of each player's control over the protocol game.

3.2.1 Fairness

Fairness is the basic symmetry property of an exchange protocol. There is a known impossibility result [18,31] demonstrating that no deterministic two-party protocol can be fair. Therefore, fairness requires introduction of at least one other party, *e.g.*, the trusted third party T . Our definition is equivalent to a common definition of fairness in terms of state reachability [22,14]. Intuitively, a protocol is fair for the honest signer A , if, whenever B has obtained A 's signature, A has a strategy in coalition with T to obtain B 's signature.

Definition 8 *A protocol is fair for honest A if, for each state S reachable by honest A such that B possesses A 's signature in S , the coalition of A and T has an honest strategy from S to give A B 's signature.*

In the remainder of this section, we show that this definition is equivalent to the standard definition of fairness in terms of state reachability.

Definition 9 *A state S reachable by honest A is potentially successful for A if there is a finite trace tr from S terminating in a state in which A has B 's signature and each transition rule in tr is labeled by a rule in $\mathbf{AUTUA}_{\text{timeouts}}$.*

Note that the existence of such a trace does not mean that A can *always* obtain B 's signature regardless of what B does.

We now show that B 's timers do not affect whether a state is potentially successful for A . The intuitive reason for this is the observation that the actions of A and T do not depend on the state of B 's timers. Therefore, timeouts of B do not affect A 's ability to contact T and obtain B 's signature even if B has succeeded in obtaining A 's signature.

Proposition 10 *Let S, S' be reachable states such that S' is obtained from S by an application of $t_1 \in \mathbf{B}_{\text{timeouts}}$ followed by an application of $t_2 \in \mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T}$. We can commute the order of application of t_1 and t_2 , i.e., S' can also be obtained from S by an application of t_2 , followed by an application of t_1 .*

Proof: Timer predicates of B do not occur in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T}$. Therefore, t_1 and t_2 affect independent parts of S and by proposition 1, we can commute the order of application of t_1 and t_2 . \square

Proposition 11 *Suppose there is a trace tr from S that uses only transition rules in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T} \cup \mathbf{B}_{\text{timeouts}}$ and terminates in a state in which A has B 's signature. Then S is potentially successful for A .*

Proof: By inductively applying proposition 10 to trace tr , we push the timeouts of B towards the end of the trace and obtain trace tr' from S which
(i) ends in a state in which A has B 's signature, and
(ii) uses only transition rules in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T}$ followed by timeout rules of B 's timers.

Timeouts of B do not affect terms in A 's possession. We conclude that tr' is a trace from S that uses only transition rules in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T}$ and ends in a state in which A has B 's signature. Hence, S is potentially successful for A . \square

We now state fairness in terms of reachability and show the equivalence of the two definitions.

Proposition 12 *A protocol is fair for honest A if and only if, for all states S reachable by honest A such that B has A 's signature on the pre-agreed text in S , the state S is potentially successful for A .*

Proof: (\Rightarrow) Intuitively, if B quits the protocol after having received A 's signature, then a fair protocol must provide some means for A to get B 's signature. This may involve contacting T . In particular, if B has A 's signature in state S , there must be a trace from S that involves only A , T and timeouts of A leading to a state in which A has B 's signature. Hence, there any state in which B has A 's signature is potentially successful for A .

Suppose that the protocol is fair for honest A , and S is a state reachable by honest A such that B has A 's signature in S . There are two possibilities: either B is still participating in the protocol (B_i occurs in S for some role state B_i) or B has dishonestly quit the protocol ($B_{dquit,i}$ occurs in S).

Consider the former case in which B has already quit the protocol in S , and let ctr be the tree of all possible traces at S . Now, since B has quit, each edge in ctr must be labeled by a rule in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{B}_{\text{timeouts}}$. The coalition of A and T controls all edges labeled by rules in $\mathbf{A} \cup \mathbf{T}$. If E is a selection of edges under the control of A and T , then, by definition, $ctr \setminus E$ is a strategy for the coalition of A and T .

If the protocol is fair, then there must be a strategy for the coalition of A and T to give A the signature of B . Therefore, there exists at least one selection of edges E such that A has B 's signature in every leaf node of the tree $ctr \setminus E$. Pick one such selection E , and any leaf node of $ctr \setminus E$. Consider the trace from the root to the chosen leaf. This trace ends in a state in which A has B 's signature and each transition is labeled by a rule in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{B}_{\text{timeouts}}$. By proposition 11, S is potentially successful for A .

If however, B has not quit the protocol in S , then let S^* be the state obtained from S using the rule of dishonest quitting in $\mathbf{B}_{\text{threat}}$. In S^* , B still has A 's signature (possession is monotonic), and hence by what we just proved, S^* must be potentially successful for A . This means that there is a trace, tr from S^* using just the transitions in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T}$ leading to a state in which A has B 's signature.

The quitting rule in B_{threat} uses just the internal role states of B and B 's dishonest memory. These predicates do not occur in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T}$. Hence, by repeated application of proposition 1, we can commute the transition from S to S^* with the whole trace tr . In this way, we obtain a trace where all edges are transitions in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{T}$, and the trace ends in a state in which A has B 's signature. Therefore, S is potentially successful for A .

(\Leftarrow) Intuitively, once B has obtained A 's signature, he continues to possess it in all subsequent states. Since every such state is potentially successful for A , the coalition of A and T may safely perform any of the actions available to them. Any state from which there are no possible continuations must also be potentially successful for A . Since there are no further actions, A must have B 's signature in that state.

Suppose S is a state reachable by honest A such that B has A 's signature in S . Let ctr be the continuation tree at S . The coalition of A and T controls all edges labeled by rules in $\mathbf{A} \cup \mathbf{T}$. To prove that the protocol is fair, we need to show that there is a selection E of edges under A 's control such that in each leaf node of $ctr \setminus E$, A has B 's signature.

Let E be an empty set and consider the strategy $ctr \setminus E = ctr$. Pick a leaf node N in the strategy and fix it. Let N be labeled by state S' . Since B has A 's signature in S and possession is monotonic, B has A 's signature in S' also. Therefore, S' is potentially successful, and there is a trace from S' with edges labeled by rules from $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A}_{\text{timeouts}}$ leading to a state in which A has B 's signature. Since ctr is the continuation tree, and N is a leaf node, there are no further traces from S' . Therefore, A must have B 's signature in S' . Since N is an arbitrary leaf node, ctr must be a strategy of the coalition of A and T to give A B 's signature. \square

3.2.2 Advantage

Intuitively, fairness says that either both players obtain what they want, or neither does. This is not always sufficient, however. A player's ability to decide unilaterally *whether* the transaction happens or not can be of great value in scenarios where resource commitment is important, such as online trading and auction bidding.

To characterize the degree to which each participant controls the outcome of the protocol in a given state, we define a pair of *resolve functions* $rslv_A, rslv_B$ which associates each reachable state with a value in $\{0, \frac{1}{2}, 1, 2\}$. We are interested in what a participant *may* do if his opponent quits the protocol. Therefore, despite our assumption that A is honest, we will consider A 's dishonest moves, including control over A 's own (but not B 's) timers, when reasoning about A 's ability to control the outcome. Intuitively, our assumption that A is honest is equivalent to stating that A follows the protocol specification, while values of the $rslv_A$ function characterize all potential outcomes if B quits the protocol, which may involve A making a dishonest move.

Definition 13 *Define the resolve function $rslv_A$ for any reachable state S as follows:*

$$\begin{aligned}
 rslv_A(S) &= 2, && \text{if } A \text{ has a strategy to obtain } B \text{'s signature,} \\
 &= 1, && \text{if } rslv_A(S) \neq 2, \text{ but } A \text{ has a } B \text{-silent strategy} \\
 &&& \text{to reach state } S' \text{ such that } rslv_A(S') = 2, \\
 &= \frac{1}{2}, && \text{if } rslv_A(S) \neq \{1, 2\}, \text{ but there is state } S' \text{ reach-} \\
 &&& \text{able from } S \text{ such that } rslv_A(S') = 2, \text{ and no} \\
 &&& \text{transition on the } S \rightarrow S' \text{ path is in } \mathbf{B} \cup \mathbf{B}_{\text{threat}}, \\
 &= 0, && \text{otherwise.}
 \end{aligned}$$

The strategies need not be honest. The definition of $rslv_B$ is symmetric.

Intuitively, $rslv_A(S) = 2$ if A can obtain B 's signature no matter what B does,

1 if A can obtain B 's signature provided B stops communicating and remains silent, $\frac{1}{2}$ if there is a possibility (but no strategy) for A to obtain B 's signature when B is silent, and 0 means that A cannot obtain B 's signature without B 's involvement. The difference between 1 and $\frac{1}{2}$ is essential. For example, $rslv_A(S) = 1$ if A can obtain B 's signature by sending a message to T as long as B is silent, while $rslv_A(S) = \frac{1}{2}$ if B is silent, but some previously sent message is already on the channel to T , and the outcome of the protocol depends on the race condition between this message and A 's message.

Given an initial state S_0 , if $rslv_A(S_0) \neq 0$ then there is a possibility for A to obtain B 's signature without B ever participating in the protocol. We believe that this is not meaningful because A might get B 's signature without B ever indicating its willingness for the exchange. For this reason, we shall assume that $rslv_A(S_0) = 0$. Similarly, we assume that $rslv_B(S_0) = 0$.

Definition 14 B has an abort strategy in S if B has a strategy to reach a state S' such that $rslv_A(S') = 0$. B has a resolve strategy in S if B has a strategy to reach a state S'' such that $rslv_B(S'') = 2$. B has an advantage in S if B has both an abort strategy and a resolve strategy.

It follows directly from definition 14 that if B has an advantage in S , then A does not have an advantage in S , and vice versa.

3.2.3 Optimism

Intuitively, a protocol is optimistic if it enables two honest parties to exchange signatures without involving the trusted third party, assuming they do not time out waiting for each other's messages. Such protocols potentially provide a practical means of fair exchange between mistrusting agents without relying on a third party in most instances.

Let S, S' be reachable states such that S' is obtained from S by a transition in $\mathbf{B} \cup \mathbf{B}_{\text{threat}}$. We say that B sends a message to T in this transition if and only if a fact created by this transition matches a term in the left hand side of a rule in \mathbf{T} .

Definition 15 A fair protocol is optimistic for B if

- (1) If S, S' are reachable states such that S is obtained from S' by a transition in \mathbf{B} , then honest B sends a message to T in this transition only if $Z(k_b, \text{timed_out}) \in S$ for some timer Z of B .
- (2) If A is honest and B controls the timeouts of both A and B , B has an honest strategy at S_0 such that
 - All edges are labeled by transitions in $\mathbf{A} \cup \mathbf{B}$.
 - Every leaf node is labeled by a state in which B possesses A 's signature.

Any trace in this strategy is an optimistic trace. The definition of optimistic for A is symmetric. A protocol is optimistic if it is optimistic for both signers.

Intuitively, the first condition implies that the protocol specification does not permit honest signers to contact T nondeterministically, *i.e.*, an honest signer only contacts T after a timeout of some timer. Also, since the strategy mentioned in the second condition is from the initial state and contains no timeouts, neither signer sends any messages to T while following an optimistic trace. Therefore, our definition of optimism implies that the signers can complete the exchange without involving T .

3.2.4 Timeliness

We now formalize the following intuition: “one player cannot force the other to wait for any length of time — a fair and timely termination can always be forced by contacting the third party” [4]. Timeliness has been emphasized by the designers of fair exchange protocols, since it is essential for practical use. In any state of the protocol, each participant should be able to terminate the exchange unilaterally. If he has not been able to obtain the other’s signature, he can always reach a terminal state where he may stop and be sure that the opponent will not be able to obtain his signature, either.

Definition 16 A fair, optimistic protocol is timely for B if in every state on an optimistic trace B has an A -silent strategy to reach a state S' such that $rslv_A(S') = 0$ or $rslv_B(S') = 2$. A protocol is timely if it is timely for both signers.

To illustrate the importance of timeliness, consider a protocol that is *not* timely, *e.g.*, the Boyd-Foo protocol [9]. In this protocol, originator O releases some information that can be used by responder R to obtain O ’s signature from T at some later point. If R stops communicating, O is at his mercy. He may have to wait, possibly forever, before he learns whether the exchange has been successful.

For the rest of this paper, we assume that the protocol is fair, timely, and optimistic for both signers.

4 Impossibility of Balance in Optimistic Protocols

As explained in the introduction, optimistic contract signing protocols are only valuable insofar as they offer benefit to an optimistic participant. We say that the honest participant A is *optimistic* if, in any state where he is permitted

by the protocol specification to contact trusted third party T , he waits for B 's response before contacting T .

For example, consider the Garay-Jakobsson-MacKenzie contract signing protocol [22]. The protocol starts with O sending his *designated-verifier signature* to R , and R responding with his own designated-verifier signature. While the protocol specification permits R to contact T immediately with a resolve request, in reality R is likely to be optimistic, *i.e.*, he will prefer to resolve the protocol amicably by normal exchange with O instead of resorting to T as soon as he has an opportunity to do so. Therefore, after sending his designated-verifier signature to O in the second message of the protocol, R will wait for O 's response for a relatively long time before contacting T . Since O has the ability to contact T with an abort request while R is waiting, at this point in the protocol O enjoys advantage against R .

As this example demonstrates, the propensity of the optimistic participant to wait for the opponent's response before contacting T can be exploited by the opponent. Recall that definition 15 implies that an honest participant only contacts T after some timer times out. We use this to model optimism of A by giving B the ability to schedule the timeout rules of A by an "out-of-band" signal. In any implementation of the protocol, B does not actually schedule A 's timers. This is simply a technical device to restrict the set of execution traces under consideration to those that may occur when one of the participants is optimistic.

Definition 14 can thus be extended to cases where A is optimistic by permitting B 's strategy to include control over timeouts of both A and B . This leads us to the following protocol property:

Definition 17 *If B does not have a strategy for reaching a state S where B has an advantage against an optimistic A , the protocol is balanced for an optimistic A .*

If a protocol is balanced, then the optimism of a signer cannot be exploited by a dishonest counterparty. As we will now show, balance cannot be achieved by any fair, timely, optimistic protocol. Before we plunge into the details of the proof, it is worth giving an informal summary. We consider the protocol from the viewpoint of the dishonest signer B . Timeliness requires that B has an abort strategy available in the beginning of the protocol so that he can terminate the protocol if A quits early. This strategy may involve B contacting the trusted third party T . As long as A continues the normal execution of the protocol, he does not contact T (this follows from the fact that the protocol is optimistic), which implies that the abort strategy remains available to B . In order for signature exchange to be successful, at some point in the normal execution of the protocol A must send a message to B that gives B the ability

to obtain A 's signature. This is precisely the point where B has both the strategy to obtain A 's signature *and* the strategy to abort the exchange. The main part of the proof is formally identifying this point, and proving that it exists in *any* fair, timely, optimistic protocol

The first observation underlying our proof is that, in the interleaving semantics of concurrency used by our model, the order of application of state transition rules that affect independent parts of the system can be commuted. The second observation is that the strategies available to the dishonest player are not negatively affected by messages sent to him by the honest player or by the honest player's timeouts because the dishonest player is free to ignore both. We start with an auxiliary proposition, which follows directly from definition 13.

Proposition 18 *If $rslv_A(S) > 0$, then there exists a trace from S to S' such that $rslv_A(S') = 2$ and no transition in this trace is in $\mathbf{B} \cup \mathbf{B}_{\text{threat}}$.*

Proof: If $rslv_A(S) = 2$, let $S' = S$ and the trace is empty. If $rslv_A(S) = 1$, pick any path in the tree corresponding to A 's B -silent strategy for reaching S' such that $rslv_A(S') = 2$. If $rslv_A(S) = \frac{1}{2}$, follows immediately from the definition. \square

Proposition 19 *Let $S \rightarrow S'$ be a state transition not in $\mathbf{B} \cup \mathbf{B}_{\text{threat}}$. If $rslv_B(S) = 2$, then $rslv_B(S') = 2$. If $rslv_A(S) = 0$, then $rslv_A(S') = 0$.*

Proof: If $rslv_B(S) = 2$ then, by definition of a strategy, in any state S' obtained by a transition not under control of B , B must also have a strategy.

Suppose $rslv_A(S) = 0$ and $rslv_A(S') > 0$. By proposition 18, there exists a trace from S' to S'' such that $rslv_A(S'') = 2$ and no transition in the trace is in $\mathbf{B} \cup \mathbf{B}_{\text{threat}}$. Prepending the $S \rightarrow S'$ transition to the path, we obtain a trace from S to S'' such that $rslv_A(S'') = 2$ and no transition on the trace is in $\mathbf{B} \cup \mathbf{B}_{\text{threat}}$. Therefore, $rslv_A(S) \geq \frac{1}{2}$, which contradicts our assumption. \square

Proposition 19 implies that if $S \rightarrow S'$ is a transition in an optimistic trace such that $rslv_A(S) = 0$ and $rslv_A(S') > 0$, then it must be in $\mathbf{B} \cup \mathbf{B}_{\text{threat}}$. Similarly, if $rslv_B(S) = 0$ and $rslv_B(S') > 0$, then $S \rightarrow S'$ is in $\mathbf{A} \cup \mathbf{A}_{\text{threat}}$. Intuitively, a player acquires some degree of control over the outcome of the protocol for the first time only because of the other player's move.

Since a timeout does not affect possession and a (potentially) dishonest A can always ignore the state of timers, the following proposition holds. (The proof has been omitted for space considerations, and is available at the ftp-site ftp://ftp.cis.upenn.edu/pub/papers/scedrov/cmss_optimjlap.pdf.)

Proposition 20 *Suppose S' is obtained from S by a rule from $\mathbf{A}_{\text{timeouts}}$.*

Then $rslv_A(S') = rslv_A(S)$.

Just like we defined $ctr_{[A]}$ to be the tree obtained from ctr by removing all edges in $\mathbf{A} \cup \mathbf{A}_{\text{threat}}$, we define $ctr_{[A+]}$ to be the tree obtained from ctr by removing all edges in $\mathbf{A} \cup \mathbf{A}_{\text{threat}} \cup \mathbf{A}_{\text{timeouts}}$. If E is a selection of edges in $ctr_{[A+]}$ under B 's control, then $ctr_{[A+]}\setminus E$ is a strategy available to B if A remains silent *and* no timers time out. We will call such a strategy *weak A-silent strategy*.

Proposition 21 *Let $S \rightarrow S'$ be a state transition in $\mathbf{A}_{\text{timeouts}}$. B has a weak A-silent abort [resolve] strategy at S' if and only if B has a weak A-silent abort [resolve] strategy at S .*

Proof: The proof depends on the observation that the actions of B and T are independent of timeouts of A , and the fact that a timeout of A does not change $rslv_A$. Hence a weak A-silent abort [resolve] strategy at S can be mimicked at S' and vice-versa.

(\Rightarrow) We show that if B has a weak A-silent abort [resolve] strategy at S , then it also has a weak A-silent abort [resolve] strategy at S' by induction on the height of continuation tree at S .

Base case: The height of the continuation tree at S is 0. Then there are no states reachable from S and the proposition is vacuously true.

Induction hypothesis: Suppose the proposition is true for all reachable states S such that the height of the continuation tree at S is $\leq n$.

Induction step: Now consider a reachable state S such that (1) the continuation tree at S has height $n+1$, and (2) B has a weak A-silent abort [resolve] strategy at S . Fix the weak A-silent abort [resolve] strategy at S . Let S' be the state obtained from S using a state transition in $\mathbf{A}_{\text{timeouts}}$. We have to show that B has a weak A-silent strategy at S' . Call the continuation tree at S' ctr' .

Consider the edges in ctr' coming out of the root. Remove all edges that are transitions in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}}$. Each of the remaining edges, if any, is a transition in $\mathbf{B} \cup \mathbf{B}_{\text{timeouts}} \cup \mathbf{T}$. It is easy to see that any such transition can also be applied at S . For each remaining edge e do the following:

Let S'' be the state obtained as a result of e . If a corresponding transition is not present in the weak A-silent abort [resolve] strategy at S , then remove this edge along with all of its descendants. If the transition is present in the weak A-silent abort [resolve] strategy, then let S_1 be the state obtained by applying this transition to S . We obtain that

- (i) the height of the continuation tree at S_1 is $\leq n$,
- (ii) B has a weak A-silent abort [resolve] strategy at S_1 ,

(iii) S'' can be obtained from S_1 by a transition in $\mathbf{A}_{\text{timeouts}}$.

By applying the induction hypothesis to S_1 , we conclude that B has a weak A -silent abort [resolve] strategy at S'' . Replace the continuation tree at S'' by this strategy. These operations produce a weak A -silent strategy for B at S' . There are two cases:

Case 1: The height of this strategy is 0. It follows from construction that the height of the weak A -silent abort [resolve] strategy at S is also 0. Therefore, $rslv_A(S) = 0$ [$rslv_A(S) = 1$]. By proposition 20, $rslv_A(S') = 0$ [$rslv_A(S') = 1$].

Case 2: The height of this strategy is ≥ 0 . By construction, all leaf nodes are labeled by states S''' such that $rslv_A(S''') = 0$ [$rslv_A(S''') = 1$]. Therefore, B has a weak A -silent abort strategy at S' . This completes the induction.

(\Leftarrow) Suppose B has a weak A -silent abort [resolve] strategy at S' . We prove by induction on the height of the continuation tree at S that B also has a weak A -silent abort strategy at S .

Base case: The height of the continuation tree at S is 0. Then there are no states reachable from S and the proposition is vacuously true.

Induction hypothesis: Suppose the lemma is true for all reachable states S such that the height of the continuation tree at S is $\leq n$.

Induction step: Consider a reachable state S such that (1) the continuation tree at S has height $n + 1$, (2) S' is obtained from S by a transition in $\mathbf{A}_{\text{timeouts}}$, and (3) B has a weak A -silent abort [resolve] strategy at S' . Fix the weak A -silent abort [resolve] strategy at S' . Let ctr be the continuation tree at S .

Consider the edges in ctr coming out of the root. Remove all edges that are labeled by transitions in $\mathbf{A} \cup \mathbf{A}_{\text{timeouts}}$ along with their descendants. Each remaining edge, if any, is a transition in $\mathbf{B} \cup \mathbf{B}_{\text{timeouts}} \cup \mathbf{T}$. It is easy to see that any such transition can also be applied at S' . For each remaining edge e do the following:

Let S_1 be the state obtained as a result of e . If a corresponding transition is not present in the weak A -silent abort [resolve] strategy at S' , then remove this edge along with all descendants. If a corresponding transition is present in the weak A -silent abort [resolve] strategy, then let S'' be the state obtained by applying this transition to S' . We obtain that

- (i) the height of the continuation tree at S_1 is $\leq n$,
- (ii) S'' can be obtained from S_1 by a transition in $\mathbf{A}_{\text{timeouts}}$,
- (iii) B has a weak A -silent abort [resolve] strategy at S'' .

By applying the induction hypothesis to S_1 , we conclude that B has a weak

A -silent abort [resolve] strategy at S_1 . Replace the continuation tree at S_1 by this strategy. These operations produce a weak A -silent strategy for B at S . There are two cases:

Case 1: The height of this strategy is 0. It follows from the construction that the height of the weak A -silent abort [resolve] strategy at S' is also 0. Therefore, $rslv_A(S') = 0$ [$rslv_A(S') = 1$]. By proposition 20, $rslv_A(S) = 0$ [$rslv_A(S) = 1$].

Case 2: The height of this strategy is ≥ 0 . By construction, all leaf nodes are labeled by states S_2 such that $rslv_A(S_2) = 0$ [$rslv_A(S_2) = 1$]. Therefore, B has a weak A -silent abort [resolve] strategy at S . This completes the induction. \square

We now establish that the strategies available to the dishonest player are not negatively affected by the honest player's timeouts.

Proposition 22 *B has an A -silent abort [resolve] strategy at S if and only if B has a weak A -silent abort [resolve] strategy at S .*

Proof: By proposition 21, a timeout of A does not affect the existence of a weak A -silent abort strategy. We use this fact to construct A -silent abort [resolve] strategies from weak A -silent [resolve] strategies by induction on the height of continuation trees. Similarly, we construct weak A -silent abort [resolve] strategies from A -silent abort [resolve] strategies by induction.

(\Rightarrow) We start by proving that if B has an A -silent strategy, then B has a weak A -silent strategy. The proof is by induction over the height of the continuation tree at S .

Base case: The height of the continuation tree at S is 0. Therefore, no state can be obtained from S . The proposition is vacuously true.

Induction hypothesis: Suppose the proposition is true for any state S such that the height of the continuation tree at S is $\leq n$.

Induction step: Consider state S such that (1) the continuation tree at S has height $n + 1$, and (2) B has an A -silent strategy at S . Let ctr be the continuation tree at S (with all transitions in $\mathbf{A} \cup \mathbf{A}_{\text{threat}}$ removed). Fix the A -silent strategy $ctr \setminus E$ at S and consider the edges coming out of its root. There are two cases:

Case 1: There is an edge labeled by a transition in $\mathbf{A}_{\text{timeouts}}$ leading to state S' . Then, by definition of the A -silent strategy, B must have an A -silent strategy at S' . By the induction hypothesis, B also has a weak A -silent strategy at S' . By proposition 21, B has a weak A -silent strategy at S .

Case 2: No edges are labeled by a transition in $\mathbf{A}_{\text{timeouts}}$. Then remove all edges of ctr originating from the root that are not present in $ctr \setminus E$. Call this tree ctr' . If no edges remain, then $rslv_A(S) = 0$ [$rslv_B(S) = 2$] and we are done. If there are some remaining edges, then for every child of the root in ctr' tree, B must have an A -silent strategy. Hence if S' is a child of the root in S' , then we can use the induction hypothesis to replace the continuation trees at S' by a weak A -silent strategy.

(\Leftarrow) We now show that if B has a weak A -silent strategy, then B has an A -silent strategy. The proof is by induction on the height of the continuation tree at S .

Base case: The height of the continuation tree at S is 0. Therefore, no state can be obtained from S . The proposition is vacuously true.

Induction hypothesis: Suppose the proposition is true for any state S such that the height of continuation tree at S is $\leq n$.

Induction step: Now consider state S such that (1) the continuation tree at S has height $n + 1$, and (2) B has a weak A -silent strategy at S . Let ctr be the continuation tree at S . Fix the weak A -silent abort strategy $ctr \setminus E$ at S and consider the edges coming out of its root. Remove all edges that are transitions in $\mathbf{A} \cup \mathbf{A}_{\text{threat}}$ along with their descendants. For each remaining edge e , perform the following operations:

Case 1: If e is a transition in \mathbf{T} leading to state S' , then the strategy $ctr \setminus E$ must also contain this transition. Hence B has a weak A -silent strategy at S' . By the induction hypothesis, B has a A -silent strategy at S' . Replace the continuation tree at S' by this strategy.

Case 2: If e is a transition in $\mathbf{B} \cup \mathbf{B}_{\text{timeouts}}$ leading to state S' , then, as in case 1, if this edge is part of the strategy at S , replace the continuation tree at S' by an A -silent strategy. If e is not part of the strategy, then remove this edge along with its descendants.

Case 3: If e is a transition in $\mathbf{A}_{\text{timeouts}}$ resulting in state S' , then, by proposition 21, B has a weak A -silent strategy at S . By the induction hypothesis, replace the continuation tree at S' by an A -silent strategy.

These operations produce an A -silent strategy for B at S' . There are two possibilities:

Case A: The height of this strategy is 0. It follows from the construction that the height of the weak A -silent strategy at S is also 0. Hence $rslv_A(S) = 0$ [$rslv_B(S) = 2$] and we are done.

Case B: The height of this strategy is > 0 . Then, by construction, all leaf nodes are labeled by states S'' such that $rslv_A(S'') = 0$ [$rslv_B(S'') = 2$].

Therefore, by induction, B has an A -silent abort [resolve] strategy at S . \square

We now show that the strategies available to dishonest players are not negatively affected by the honest player's messages to the dishonest player.

Lemma 23 *Let $S \rightarrow S'$ be a transition in $\mathbf{A} \cup \mathbf{A}_{\text{threat}}$. If B has an A -silent abort [resolve] strategy in S , and A does not send a message to T in the $S \rightarrow S'$ transition, then B has an A -silent abort [resolve] strategy in S' .*

Proof: We rely on the observation that state transition rules affecting independent parts of the system may be commuted. Intuitively, moves of B and T are independent of A 's internal state. Therefore, as long as A does not send any messages to T , B may ignore any message sent to him by A and follow the same strategy in S' as in S . In light of proposition 22, all we need to show is that B has a weak A -silent abort [resolve] strategy at S' if B has a weak A -silent abort [resolve] strategy at S . We prove this by induction on the height of the continuation tree at S .

Base case: The height of the continuation tree at S is 0. The lemma is vacuously true.

Induction hypothesis: Suppose the lemma is true for all states S such that the height of the continuation tree at S is $\leq n$.

Induction step: Consider state S such that (i) the height of the continuation tree at S is $n + 1$, and (ii) B has a weak A -silent abort [resolve] strategy at S .

Consider the continuation tree at S' , and remove all edges that are in $\mathbf{A} \cup \mathbf{A}_{\text{threat}} \cup \mathbf{A}_{\text{timeouts}}$ along with their descendants. For each remaining edge e from S' to some state S'' , let t be the state transition rule labeling e and consider the following cases:

Case 1: $t \in \mathbf{T}$. Since no message is sent to T in the $S \rightarrow S'$ transition, t can be applied at S as well, resulting in some state \hat{S} . Observe that

- (i) the height of the continuation tree at \hat{S} is $\leq n$,
- (ii) B has a weak A -silent strategy at \hat{S} ,
- (iii) S'' can be obtained from \hat{S} by the same transition that labels $S \rightarrow S'$.

All that is needed is to commute $S \rightarrow S'$ and $S' \rightarrow S''$ transitions.

By the induction hypothesis, B has a weak A -silent strategy at S'' . Replace the continuation tree at S'' by this strategy.

Case 2: $t \in \mathbf{B} \cup \mathbf{B}_{\text{threat}}$. There are three possibilities:

- (2.1) t cannot be applied at S . Remove edge e along with its descendants.
- (2.2) t can be applied at S , but it is not a part of the A -silent strategy at S . Remove edge e along with its descendants.
- (2.3) t can be applied at S , and it is a part of the A -silent strategy at S . Then, as in Case 1, replace the continuation tree at S'' by this strategy.

Case 3: $t \in \mathbf{B}_{\text{timeouts}}$. If t is not a part of the A -silent strategy at S , remove edge e along with its descendants. If it is a part of the A -silent strategy, replace the continuation tree at S'' by this strategy.

By constructing the right continuation tree for any immediate descendant of S' , we have constructed a weak A -silent strategy at S' . It remains to show that it is indeed an abort [resolve] strategy. There are two possibilities:

Case A: The height of the constructed strategy is 0. From the construction, it follows that the height of the weak A -silent abort [resolve] strategy at S is also 0. Therefore, $rslv_A(S) = 0$ [$rslv_B(S) = 2$]. By proposition 19, $rslv_A(S') = 0$ [$rslv_B(S') = 2$].

Case B: The height of the constructed strategy is > 0 . By construction, all leaf nodes are labeled by states S^* such that $rslv_A(S^*) = 0$ [$rslv_B(S^*) = 2$].

We conclude that B has a weak A -silent abort [resolve] strategy at S' , which completes the induction. \square

We now use lemma 23 to show that for each strategy conditional on A remaining silent, there is an equivalent unconditional strategy against an optimistic A . The reason is that an optimistic A prefers to wait for B 's messages instead of trying to contact T . Therefore, B may ignore optimistic A 's messages and proceed with the A -silent strategy, obtaining the desired protocol outcome.

Lemma 24 *Let S be a reachable state that does not contain $Z(k_a, \text{timed_out})$ for any timer predicate Z . If B has an A -silent abort [resolve] strategy in state S , then B has an abort [resolve] strategy against optimistic A in S .*

Proof: Since B has an A -silent abort [resolve] strategy at S , B has a weak A -silent abort [resolve] strategy at S by proposition 22. By lemma 23, if A does not send a message to T in a transition from S , then B will still have the weak A -silent abort [resolve] strategy in the resulting state. B can prevent an optimistic A from contacting T by controlling A 's timeouts. Using these facts, we show that B has an abort [resolve] strategy against an optimistic A which works by ignoring messages from A and preventing A from contacting T .

In particular, we show that if S is a reachable state such that (i) S does not

contain $Z(k_a, \text{timed_out})$ for any timer predicate Z , and (ii) B has a weak A -silent abort [resolve] strategy against an optimistic A , then B also has an abort strategy against an optimistic A . The proof is by induction on the height of the continuation tree at S .

Base case: The height of the tree is 0. By definition, the continuation tree ctr is simply the root labeled by S . The only possible weak A -silent strategy for B is ctr which is also a strategy against an optimistic A .

Induction hypothesis: Suppose the lemma is true for all states S such that the height of the continuation tree at S is $\leq n$.

Induction step: Consider state S such that (i) B has a weak A -silent abort [resolve] strategy at S , (ii) S does not contain any $Z(k_a, \text{timed_out})$ for any timer predicate Z , and (iii) the height of the continuation tree at S is $n+1$. Fix the weak A -silent strategy, and let ctr be the continuation tree at S . Consider the edges in ctr coming out of the root. Because A is optimistic, there are no edges labeled by transitions in $\mathbf{A}_{\text{threat}}$. For each edge e , perform the following operations:

Case 1: Edge e is a transition in \mathbf{A} . Since S does not contain any timer predicate $Z(k_a, \text{timed_out})$ and the protocol is optimistic for A , A does not send any messages to T in this transition. If the resulting state is S' , then by lemma 23 and proposition 21, B has a weak A -silent abort [resolve] strategy at S' . Since the height of continuation tree at S' is $\leq n$, B also has a abort [resolve] strategy against optimistic B at S' by the induction hypothesis. Replace the continuation tree at S' by this strategy.

Case 2: Edge e is a transition in $\mathbf{B} \cup \mathbf{T} \cup \mathbf{B}_{\text{timeouts}} \cup \mathbf{A}_{\text{timeouts}}$. Let the resulting state be S' . There are two possibilities:

(2.1) This transition is not in the weak A -silent strategy at S . Then edge e must be labeled by a transition in $\mathbf{B} \cup \mathbf{B}_{\text{timeouts}} \cup \mathbf{A}_{\text{timeouts}}$. Therefore, B controls this edge when playing against an optimistic A . Remove this edge along with its descendants.

(2.2) This transition is in the weak A -silent strategy at S . Then B also has a weak A -silent strategy at S' . By definition of weak A -silent strategies, e cannot be labeled by a transition in $\mathbf{A}_{\text{timeouts}}$. Therefore, S' does not contain $Z(k_a, \text{timed_out})$. Since the height of the continuation tree at S' is $\leq n$, the induction hypothesis applies, and we conclude that B has a strategy at S' . Replace the continuation tree at S' by this strategy.

These operations produce a strategy for B at S against an optimistic A . There are two cases:

Case A: The height of this strategy is 0. It follows from the construction that the height of the weak A -silent strategy at S is also 0. Therefore, $rslv_A(S) = 0$

$[rslv_B(S) = 2]$ and this strategy is an abort [resolve] strategy.

Case B: The height of this strategy is > 0 . By construction, all leaf nodes are labeled by states S'' such that $rslv_A(S'') = 0$ [$rslv_B(S'') = 2$]. Therefore, the strategy is an abort [resolve] strategy.

We conclude that B has an abort [resolve] strategy against an optimistic A .
 \square

To show our main impossibility result, we need one more proposition.

Proposition 25 *Consider an optimistic protocol with initial state S_0 . Let tr be an trace in which A is optimistic such that*

- (1) *the leaf node is labeled by state S' in which B possesses A 's signature,*
- (2) *each edge in tr is labeled by a transition in $\mathbf{A} \cup \mathbf{B}$.*

Then there is a non-initial state S^ in tr in which B has an A -silent abort strategy against A .*

Proof: We have $rslv_B(S_0) = 0$ and $rslv_B(S') = 2$. Consider the first transition $S \rightarrow S^*$ on tr such that $rslv_B(S) = 0$, $rslv_B(S^*) > 0$. Proposition 19 implies that this must be a transition in $\mathbf{A} \cup \mathbf{A}_{\text{threat}}$.

By definition 16, B has an A -silent strategy at S to reach a state S'' such that $rslv_A(S'') = 0$ or $rslv_B(S'') = 2$. Since $rslv_B(S) = 0$, it must be the case that $rslv_A(S'') = 0$, thus B has an A -silent abort strategy at S . The protocol is optimistic for A , tr starts in the initial state S_0 and no timeouts occur on tr . Hence A does not send any messages to T in the transition from S to S^* . Hence, by lemma 23, B also has an A -silent abort strategy in S^* . \square

We are now ready to prove our main impossibility result: in any optimistic, fair, timely protocol (potentially dishonest) B has a strategy to reach a state where B enjoys an advantage against an optimistic A .

Theorem 26 (Impossibility of Balance) *In a fair, optimistic, timely protocol between signers A and B , if A is optimistic, then B has a strategy for reaching a non-initial state S^* such that B has an advantage against A at S^* .*

Proof: Recall that if the protocol is optimistic, then B has an honest strategy at S_0 against an optimistic A such that:

- (1) each leaf node in the strategy is labeled by a state in which B possesses A 's signature, and
- (2) all transitions in the strategy are in $\mathbf{A} \cup \mathbf{B}$.

As a consequence of proposition 25, every trace in this strategy contains a

non-initial state in which B has an A -silent abort strategy. Hence, in order to reach the state in which he has advantage, B follows this strategy until it hits the state in which he has an A -silent abort strategy. Then B lets A continue the optimistic strategy until all optimistic actions of A are exhausted, reaching some state S^* . Since no timeouts happen, A does not send any messages to T and, by repeated application of lemma 23, B still has the A -silent abort strategy in S^* . By lemma 24, B has an abort strategy against optimistic A in S^* . Since S^* is a state in B 's optimistic strategy, B also has a resolve strategy against an optimistic A in S^* . Therefore, B has an advantage against optimistic A in S^* .

More precisely, assuming A and B are honest, let ctr be the continuation tree at the initial state S_0 . By definition of the optimistic protocol, there is a selection of edges E in $\mathbf{B} \cup \mathbf{A}_{\text{timeouts}} \cup \mathbf{B}_{\text{timeouts}}$ such that in $ctr \setminus E$

- (1) each leaf node is labeled by a state in which B possesses A 's signature, and
- (2) all transitions are in $\mathbf{A} \cup \mathbf{B}$.

Observe that no timeouts happen in the entire $ctr \setminus E$ tree. This implies that $Z(k_a, \text{timed_out})$ does not occur in any state, and A does not send any messages to T in any transition.

By proposition 25, for each trace in $ctr \setminus E$ there is a non-initial state S' such that B has an A -silent abort strategy. For any trace, pick the first node (counting from the root) labeled by such a state, and let \mathcal{N} be the collection of all such nodes. By construction, every trace in $ctr \setminus E$ passes through some node N in \mathcal{N} . Moreover, two nodes from \mathcal{N} may not occur in the same trace.

For each node N in \mathcal{N} , consider the subtree of $ctr \setminus E$ rooted at N . In this subtree, remove edges labeled by \mathbf{B} along with their descendants. Observe that in this new subtree (which is still a part of $ctr \setminus E$), all edges are labeled by transitions in \mathbf{A} , and A does not send any messages to T in these transitions. Since N itself is labeled by a state in which B has an A -silent abort strategy, by repeated application of lemma 23 we obtain that B has an A -silent abort strategy in each leaf node of this subtree.

After performing this operation for each node N in \mathcal{N} , we obtain a strategy for B against an optimistic A . The reason for this is that E was a selection of edges in $\mathbf{B} \cup \mathbf{B}_{\text{timeouts}} \cup \mathbf{A}_{\text{timeouts}}$ and the edges removed by our construction are in \mathbf{B} . By construction, this new strategy is a subtree of $ctr \setminus E$. We now prove that this is the strategy required by the statement of the theorem.

Consider any leaf node S^* of this strategy. By construction, the trace from S_0 to this node passes through some N in \mathcal{N} . Therefore, B has an A -silent abort strategy in S^* . Since this node is also present in $ctr \setminus E$, $Z(k_a, \text{timed_out})$ does not occur in S^* . By lemma 24, B has an abort strategy against optimistic A at S^* . Moreover, since S^* is reached a part of the $ctr \setminus E$ strategy, B also has

a strategy to obtain optimistic A 's signature at S^* . □

We emphasize that Theorem 26 applies equally to initiator and responder. An optimistic participant is at a disadvantage *regardless* of the role he plays in the protocol. For example, in the Garay-Jakobsson-MacKenzie abuse-free contract signing protocol [22], the originator enjoys an advantage against the responder, even though the responder is the first to receive information that potentially enables him to obtain the originator's signature. More generally, an optimistic player is at a disadvantage against a malicious player, regardless of whether the optimistic player is the originator or responder in the protocol.

5 Examples

In this section, we illustrate how our main impossibility theorem applies to three optimistic contract signing protocols proposed in the literature. For each protocol, we identify the point at which a dishonest participant has advantage over an optimistic counterparty. The protocols we consider are the off-line fair payment protocol of Boyd and Foo [9], the optimistic signature exchange protocol of Asokan, Shoup, and Waidner [4] (not to be confused with the protocol of [3]), and the abuse-free contract signing protocol of Garay, Jakobsson, and MacKenzie [22]. We discuss the common structure shared by all of these protocols, and informally suggest how advantage enjoyed by each of the protocol participants decreases as message exchange progresses.

5.1 Verifiable convertible signature commitments

Fixed-round optimistic signature exchange protocols in the literature usually employ various implementations of *verifiable, convertible signature commitments*. Informally, a signature commitment is a cryptographic construction that can be used by a signer to convince his counterparty (aka *designated verifier* [28]) that the signer has computed the requested signature without releasing the signature itself. The recipient can verify the commitment, but cannot convert it into a conventional, universally verifiable signature. The creator of a signature commitment identifies the trusted third party as the designated converter [11]. Computational properties of verifiable signature commitment are such that nobody other than the designated verifier or the designated converter will be convinced of its creator's identity if shown the commitment.

A signature commitment may be converted into a universally verifiable signature by the designated converter. Typically, a third party trusted by both sign-

ers is chosen as the designated converter. The trusted third party is invoked optimistically, only if one of the parties misbehaves or if there is a communication failure. The trusted third party then uses commitments exchanged by the signers to resolve the protocol fairly.

Let $vcsc_A(m, B, T)$ abstractly denote the cryptographic primitive implementing verifiable convertible signature commitment. Its properties are as follows:

- (a) $vcsc_A(m, B, T)$ can be created only by A ;
- (b) $vcsc_A(m, B, T)$ can be verified by B , but cannot be used as a proof of A 's intentions. In the case of [4,9,22], this is true because $vcsc$ is a zero-knowledge proof which can be simulated by B ;
- (c) $vcsc_A(m, B, T)$ can be converted into a universally verifiable signature $sig_A(m)$ by T .

In the definition above, *simulation* is used in the cryptographic sense (see any standard reference on foundations of cryptography such as [24]). Very informally, B can compute a proof which is indistinguishable by any probabilistic polynomial-time algorithm from the signature commitment sent by A . Therefore, B cannot use A 's commitment in lieu of A 's signature, because no third party can determine whether it was computed by A or simulated by B . The only exception is the designated converter T , who can convert A 's commitment into an actual signature.

5.2 Generic optimistic contract signing protocol

All of the optimistic protocols we used as illustrations follow the same basic logic, modulo minor variations and optimizations. Let us call the two signers O (originator) and R (responder), and the trusted third party T . There are three subprotocols, which we will call *exchange*, *abort*, and *resolve*. In the protocol description, when a participant A sends a message msg intended for B , it will be abbreviated as $A \rightarrow B: msg$. We use pd : *protocollnstance* to uniquely identify a protocol instance.

Exchange subprotocol: The parties first exchange signature commitments with T as the designated converter, and then actual signatures. This represents the optimistic flow of the protocol. If nothing goes wrong, it results in a successful exchange of signatures without T 's involvement.

$$O \rightarrow R \quad me_1 = vcsc_O(pd, R, T)$$

$$O \leftarrow R \quad me_2 = vcsc_R(pd, O, T)$$

$$O \rightarrow R \quad me_3 = sig_O(pd)$$

$$O \leftarrow R \quad me_4 = sig_R(pd)$$

Abort subprotocol: If O does not receive me_2 in response to his first message, he has the option to time out and contact T with a request not to resolve the current instance of the protocol in the future. When T receives this abort request, it checks its permanent database of past actions. If T has not previously been requested to resolve this instance of the protocol, T marks the instance as aborted in the database and sends an abort token to O . If the instance is already marked as resolved, this means that T has previously resolved this exchange in response to an earlier request (as described below). T must have obtained both $sig_O(pd)$ and $sig_R(pd)$. The latter is then released to O . The exact formats of the abort request and the abort tokens depend on the protocol. While R is usually not allowed to abort, he is allowed to time out and quit if he does not receive me_1 .

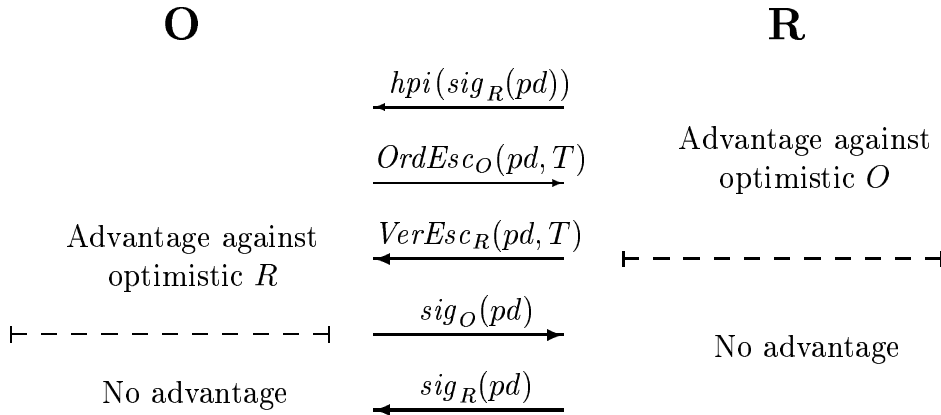
$$\begin{aligned}
O \rightarrow T & \quad ma_1 = sig_O(abort, me_1) \\
O \leftarrow T & \quad ma_2 = \text{Has } me_1 \text{ been resolved already?} \\
& \quad \text{Yes : } sig_R(pd) \\
& \quad \text{No : } sig_T(abort, me_1) \\
& \quad aborted[me_1] := \text{true}
\end{aligned}$$

Resolve subprotocol: If O sends his signature to R in me_3 , but does not receive R 's signature in return, he can appeal to T using R 's signature commitment received in me_2 . T will convert R 's commitment into a universally verifiable signature. Similarly, if R releases his commitment to O in me_2 , but does not receive O 's signature in return, he can ask T to convert O 's commitment received in me_1 into a universally verifiable signature. As part of the resolve request, the requester must release his own signature to T or send his designated converter signature to T . The exact format of the resolve request depends on the protocol.

$$\begin{aligned}
R(O) \rightarrow T & \quad mr_1 = vcsc_O(pd, R, T), vcsc_R(pd, O, T) \\
R(O) \leftarrow T & \quad mr_2 = \text{Has } me_1 \text{ been aborted already?} \\
& \quad \text{Yes : } sig_T(abort, me_1) \\
& \quad \text{No : Convert } vcsc_O(pd, R, T), vcsc_R(pd, R, T) \\
& \quad \text{into } sig_O(pd), sig_R(pd) \\
& \quad resolved[me_1] := \text{true}
\end{aligned}$$

5.3 Advantage in Asokan-Shoup-Waidner protocol

The Asokan-Shoup-Waidner optimistic protocol [4] follows the pattern of the generic protocol described in section 5.2 with one additional message flow. Prior to the main exchange subprotocol, R reduces his signature to a homomorphic pre-image (hpi) which can be verified in the same way signature is, but at the same time preserves secrecy of the signature on which it is based (see [4] for details). Also, for optimization, me_1 contains ordinary escrow of $sig_O(pd)$ instead of a verifiable commitment. Ordinary escrow $OrdEsc_O(pd, T)$ can be converted into $sig_O(pd)$ by T , but R cannot verify this independently of T . R 's response contains a verifiable, convertible signature commitment in the sense of section 5.1, implemented as a verifiable escrow $VerEsc_R(pd, T)$. O can verify independently of T that the escrow indeed contains $sig_R(pd)$ and T will be able to convert it into $sig_R(pd)$ if necessary. The following picture illustrates how advantage of each party decreases as message exchange in the exchange subprotocol progresses.



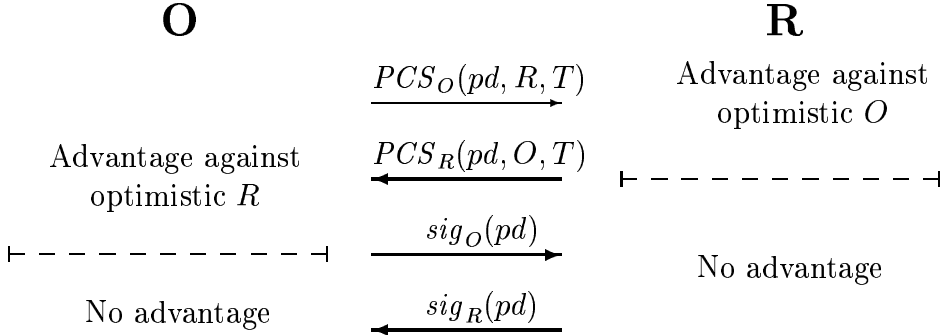
Even though O has sent $OrdEsc_O(pd, T)$ in his first message, O still has an advantage against an optimistic R until he sends out $sig_O(pd)$. This is because O can abort by contacting T and ignoring all messages from R . An optimistic R will prefer to wait for O 's response rather than contact T with a resolve request. If O wants to complete the exchange, he simply continues the exchange subprotocol. The advantage flow of R can be similarly reasoned out.

5.4 Advantage in Garay-Jakobsson-MacKenzie protocol

The abuse-free contract signing protocol of Garay, Jakobsson, and MacKenzie protocol [22] is very similar to the Asokan-Shoup-Waidner protocol. The only essential difference between the two protocols is in the details of cryptographic implementation, and is thus not reflected in our abstract model. In the ASW

protocol, verifiable convertible signature commitments are implemented via verifiable escrows, which are constructed as interactive zero-knowledge proofs of commitment to a signature. This means that B may be able to convince some outside party C that A is participating in the protocol as long as C is online and can observe the interaction between A and B .

By contrast, the GJM protocol uses *private contract signatures (PCS)*, which are *non-interactive* signature commitments. Therefore, B cannot prove to C that A is participating in the protocol. This property is known as *abuse-freeness*. Even though a dishonest participant has advantage over an optimistic counterparty in the GJM protocol, exploiting the advantage is more difficult (*e.g.*, a dishonest auctioneer cannot reveal an optimistic buyer’s bid to another potential buyer).



5.5 Advantage in Boyd-Foo protocol

We now discuss a protocol derived from the Boyd-Foo protocol [9]. The protocol uses the Gennaro-Krawczyk-Rabin (GKR) scheme [23] for designated-converter signatures. A designated-verifier extension of the scheme is also discussed in [23]. We will denote the designated-verifier, designated-converter signature from O intended for R with converter T as $S(pd, k_o, k_r, k_t)$. This can be thought of as a realization of the *vcsc* primitive discussed in section 5.1.

This protocol differs from the generic protocol in that only three messages are used in the exchange protocol. The exchange protocol starts with O sending $S(pd, k_o, k_r, k_t)$ to R , who verifies (via an interactive zero-knowledge proof) that it was indeed generated by O . R then sends back $sig_R(pd)$ to O . Finally, O sends $sig_O(pd)$ to R . Hence, the exchange subprotocol is:

$$\begin{aligned}
 O \rightarrow R & \quad me_1 = S(pd, k_o, k_r, k_t) \\
 R \rightarrow O & \quad me_2 = sig_R(pd) \\
 O \rightarrow R & \quad me_3 = sig_O(pd)
 \end{aligned}$$

7 Conclusions and Further Work

We have studied contract signing protocols in a game-theoretic model, giving precise, formal definitions of properties such as fairness and timeliness. We characterized optimism of honest protocol participants using a form of out-of-band signal that forces the optimistic player to wait for the opponent. While the out-of-band signal does not correspond to any realistic mechanism in distributed computation, it accurately reduces the set of protocol traces to those where the optimistic player waits for the opponent instead of contacting the trusted third party.

Our main result is that in any fair, optimistic, timely protocol, an optimistic player yields an advantage to his opponent. This means that the opponent has both a strategy to complete the signature exchange and a strategy to keep the players from obtaining each other's signatures. Since the protocol is fair, the outcome for both players is the same, but the player with an advantage can choose what this outcome is. This holds regardless of whether the optimistic player is the first or second mover.

Since advantage cannot be eliminated, it appears that the best a protocol can do to protect optimistic participants is prevent an opponent from proving to any outside party that he has reached a position of advantage. This property is identified in literature [22] as abuse-freeness. We are currently investigating the formalization of abuse-freeness. Another direction for further investigation involves the notion of trusted third party accountability. The relationship between our definitions and the cryptographic definitions of fairness [4] may also merit further study. Finally, we believe that our techniques may prove useful for investigating multi-party contract signing protocols.

References

- [1] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the π -calculus. *Information and Computation*, 143:1–70, 1999.
- [2] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *Proc. 4th ACM Conf. on Computer and Communications Security*, pages 7–17, 1997.
- [3] N. Asokan, V. Shoup, and M. Waidner, *Asynchronous protocols for optimistic fair exchange*, IEEE Symposium on Security and Privacy, 1998, pp. 86–99.
- [4] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.

- [5] J. Banatre and D. Le Metayer. Computing by multiset transformation. *Communications of the ACM (CACM)*, 36(1):98–111, 1993.
- [6] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.
- [7] G. Berry and D. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1992.
- [8] D. Boneh and M. Naor. Timed commitments and applications. In *Proc. CRYPTO '00*, pages 236–254, 2000.
- [9] C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. In *Proc. ASIACRYPT '98*, pages 271–285, 1998.
- [10] L. Buttyán and J.-P. Hubaux. Toward a formal model of fair exchange — a game theoretic approach. Technical Report SSC/1999/39, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, December 1999.
- [11] J. Boyar, D. Chaum, and I. B. Damgård, *Convertible undeniable signatures*, Advances in Cryptology - Proceedings of CRYPTO '90, 1990, pp. 189–205.
- [12] I. Cervesato, N.A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proc. 12th IEEE Computer Security Foundations Workshop*, pages 55–69, 1999.
- [13] I. Cervesato, N.A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. A revisited comparison between strand spaces and multiset rewriting for security protocol analysis. *Journal of Computer Security*, 2004. To Appear. Extended abstract in "Software Security – Theories and Systems. Mext-NSF-JSPS International Symposium, ISSS 2002, Tokyo, Japan, 2002, Revised Papers. Springer LNCS Volume 2609, Springer-Verlag, 2003, pages 356–383".
- [14] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract signing protocols. In *Proc. 8th ACM Conf. on Computer and Communications Security*, pages 176–185, 2001.
- [15] I. B. Damgård. Practical and provably secure release of a secret and exchange of signatures. *J. Cryptology*, 8(4):201–222, 1995.
- [16] N. Durgin and J. Mitchell. Analysis of security protocols. In *Computational System Design, Series F: Computer and Systems Sciences, Vol. 173*. IOS Press, 1999.
- [17] D. Dolev and A. Yao. On the security of public-key protocols. In *Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science*, pages 350–357, 1981.
- [18] S. Even and Y. Yacobi. Relations among public key signature schemes. Technical Report 175, Computer Science Dept. Technion, Israel, March 1980.
- [19] F.J. Thayer Fábrega, J. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. IEEE Symposium on Security and Privacy*, pages 160–171, 1998.

- [20] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
- [21] M. Fischer, N. Lynch, and M. Patterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.
- [22] J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Proc. CRYPTO '99*, pages 449–466, 1999.
- [23] R. Gennaro, T. Rabin, and H. Krawczyk. *RSA-based undeniable signatures*, *Journal of Cryptology* **13** (2000), no. 4, 397–416.
- [24] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
- [25] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.
- [26] S. Kremer and J.-F. Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *Proc. CONCUR '01*, pages 551–565, 2001.
- [27] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 206–220, 2002.
- [28] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Proc. EUROCRYPT '96*, pages 143–154, 1996.
- [29] R. Pucella and J. Halpern. Modeling adversaries in a logic for security protocol analysis. In *Formal Aspects of Security, 2002 (FASec '02)*.
- [30] O. Markowitch and S. Saeednia. Optimistic fair exchange with transparent signature recovery. In *Proc. 5th International Conf. on Financial Cryptography*, pages 339–350, 2001.
- [31] H. Pagnia and F. Gaertner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Department of Computer Science, Darmstadt University of Technology, Germany, March 1999.
- [32] T.Y.C. Woo and S.S. Lam. A semantic model for authentication protocols. In *Proc. IEEE Symposium on Security and Privacy*, pages 178–194, 1993.
- [33] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proc. IEEE Symposium on Security and Privacy*, pages 55–61, 1996.
- [34] J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In *Proc. International Refinement Workshop and Formal Methods Pacific*, pages 370–380, 1998.