Fair Exchange in Strand Spaces[*]

Joshua D. Guttman
The MITRE Corporation and
Worcester Polytechnic Institute

Many cryptographic protocols are intended to *coordinate state changes* among principals. Exchange protocols coordinate delivery of new values to the participants, e.g. additions to the set of values they possess. An exchange protocol is *fair* if it ensures that delivery of new values is balanced: If one participant obtains a new possession via the protocol, then all other participants will, too. Fair exchange requires *progress* assumptions, unlike some other protocol properties.

The strand space model is a framework for design and verification of cryptographic protocols. A *strand* is a local behavior of a single principal in a single session of a protocol. A *bundle* is a partially ordered global execution built from protocol strands and adversary activities.

The strand space model needs two additions for fair exchange protocols. First, we regard the state as a multiset of facts, and we allow strands to cause changes in this state via multiset rewriting. Second, progress assumptions stipulate that some channels are resilient—and guaranteed to deliver messages—and some principals are assumed not to stop at certain critical steps.

This method leads to proofs of correctness that cleanly separate protocol properties, such as authentication and confidentiality, from invariants governing state evolution. G. Wang's recent fair exchange protocol illustrates the approach.

1 Introduction

Many cryptographic protocols are meant to *coordinate state changes* between principals in distributed systems. For instance, electronic commerce protocols aim to coordinate state changes among a customer, a merchant, and one or more financial institutions. The financial institutions should record credits and debits against the accounts of the customer and the merchant, and these state changes should be correlated with state changes at the merchant and the customer. The merchant's state changes should include issuing a shipping order to its warehouse. The customer records a copy of the shipping order, and a receipt for the funds from its financial institution. The job of the designer of an application-level protocol like this is to ensure that these changes occur in a coordinated, transaction-like way.

State changes should occur only when the participants have taken certain actions, e.g. the customer must have authorized any funds transfer that occurs. Moreover, they should occur only when the participants have certain joint knowledge, e.g. that they all agree on the identities of the participants in the transaction, and the amount of money involved. These are *authentication* goals in the parlance of protocol analysis. There may also be *confidentiality* goals that limit joint knowledge. In our example, the customer and merchant should agree on the goods being purchased, which should not be disclosed to the bank, while the customer and bank should agree on the account number or card number, which should not be disclosed to the merchant.

Goal of this paper. In this paper, we develop a model of the interaction of protocol execution with state and state change. We use our model to provide a proof of a clever fair exchange protocol due to Guilin Wang [13], modulo a slight correction.

[*] Funded by MITRE-Sponsored Research. Email: guttman@mitre.org.

We believe that the strength of the model is evident in the proof's clean composition of protocol-specific reasoning with state-specific reasoning. In particular, our proof modularizes what it needs to know about protocol behavior into the four authentication properties given in Section 2, Lemmas 2.1–2.2. If any protocol achieves these authentication goals and its roles obey simple conditions on the ordering of events, then other details do not matter: it will succeed as a fair exchange protocol.

A two-party fair exchange protocol is a mechanism to deposit a pair of values atomically into the states of a pair of principals. Certified delivery protocols are a typical kind of fair exchange protocol. A certified delivery protocol aims to allow A, the sender of a message, to obtain a digitally signed receipt if the message is delivered to B. B should obtain the message together with signed evidence that it came from A. If a session fails, then neither principal should obtain these values. If it succeeds, then both should obtain them. The protocol goal is to cause state evolution of these participants to be *balanced*.

The "fair" in "fair exchange" refers to the balanced evolution of the state. "Fair" does not have the same sense as in some other uses in computer science, where an infinitely long execution is *fair* if any event actually occurs, assuming that it is enabled in an infinite subsequence of the states in that execution. In some frameworks, fairness in this latter sense helps to clarify the workings of fair exchange protocols [2, 5]. However, we show here how fair exchange protocols can also be understood independent of this notion of fairness. When we formalize Wang's protocol [13], we use an extension of the strand space model [10] in which there are no infinite executions or fairness assumptions.

As has been long known [8, 12], a deterministic fair exchange protocol must rely on a trusted third party T. Recent protocols generally follow [1] in using the trusted third party optimistically, i.e. T is never contacted in the extremely common case that a session terminates normally between the two participants. T is contacted only when one participant does not receive an expected message.

Each principal A, B, T has a state. T uses its state to record the sessions in which one participant has contacted it. For each such session, T remembers the outcome—whether T aborted the session or completed it successfully—so that it can deliver the same outcome to the other participant. The states of A, B simply records the ultimate result of each session in which it participates. The protocol guides the state's evolution to ensure balanced changes.

Strand space extensions. Two additions to strand spaces are needed to view protocols as solving to coordinated state change problems. A *strand* is a sequence of actions executed by a single principal in a single local session of a protocol.

We enrich strands to allow them to *synchronize* with the projection of the joint state that is local to the principal P executing the strand. We previously defined the actions on a strand to be either (1) message transmissions or (2) message receptions. We now extend the definition to allow the actions also to be *(3) state synchronization events*. P's state at a particular time may permit some state synchronization events and prohibit others, so that P's strands are blocked from the latter behaviors. Thus, the state constrains protocol behavior. Updates to P's state may record actions on P's strands.

We represent states by multisets of facts, and state change by multiset rewriting [3, 7], although with several differences from Mitchell, Scedrov et al. First, they use multiset rewriting to model protocol and communication behavior, as well as the states of the principals. We instead use strands for the protocol and communication behavior. Our multiset rewriting represents only changes to a single principal's local state. Hence, second, in our rules we do not need existentials, which they used to model selection of fresh values. Third, we tend to use "big" states that may have a high cardinality of facts. However, the big states are generally sparse, and extremely easy to implement with small data structures.

We also incorporate *guaranteed progress* assumptions into strand spaces. Protocols that establish balance properties need guaranteed progress. Since principals communicate by messages, one of them—

call it A—must be ready to make its state change first. Some principal (either A or some third party) must send a message to B to enable it to make its state change. If this message never reaches B, B cannot execute its state change. Hence, in the absence of a mechanism to ensure progress, A has a strategy—by preventing future message deliveries—to prevent the joint state from returning to balance.

These two augmentations—state synchronization events and a way to stipulate progress—fit together to form a strand space theory usable for reasoning about coordinated state change.

Structure of this paper. Section 2 describes Wang's protocol. Two lemmas (Lemmas 2.1 and 2.2) summarize the authentication properties that we will rely on. Any protocol whose message flow satisfies these two lemmas, and which synchronizes with state history at the same points, will meet our needs.

Section 3 introduces our multiset rewriting framework, proving a locality property. This property says that state synchronization events of two different principals are always concurrent in the sense that they commute. Hence, coordination between different principals can only occur by protocol messages, not directly by state changes. We also formalize the state facts and rules for Wang's protocol, inferring central facts about computations using these rules. These (very easily verified) facts are summarized in Lemma 3.7. Any system of rules that satisfies Lemma 3.7 will meet our needs.

Section 4 gives definitions for guaranteed progress, applying them to Wang's protocol. Lemma 4.4, the key conclusion of Section 4, says that any compliant principals executing a session with a session number L can always proceed to the end of a local run, assuming only that the trusted third party is "ready" to handle sessions labeled L.

In Section 5 we put the pieces together to show that it achieves its balanced state evolution goal. In particular, the balance property depends only on Lemmas 2.1 and 2.2 about the protocol structure, Lemma 3.7 about the state history mechanism, and lemma 4.4 about progress. In this way, the verification is well-factored into three sharply distinguished conceptual components.

2 The Gist of Wang's Protocol

Wang's fair exchange protocol [13] is appealing because it is short—only three messages in the main exchange (Fig. 1)—and uses only "generic" cryptography. By generic cryptography, Wang means standard digital signatures, and probabilistic asymmetric encryption such that the random parameter may be recovered when decryption occurs. RSA-OAEP is such a scheme. In many situations, these advantages will probably outweigh one additional step in the dispute resolution (see below in this section, p. 5).

We write $\{\!|t|\!\}_k$ for t encrypted with the key k, and $\{\!|t|\!\}_k^r$ for t encrypted with the key k using recoverable random value r. We write $\mathsf{h}(t)$ for a cryptographic hash of t, and $[\![t]\!]_k$ for a digital signature on t which may be verified using key k. By this, we mean t together with a cryptographic value prepared from $\mathsf{h}(t)$ using k^{-1}, the private signature key corresponding to k. When we use a principal name A, B, T in place of k, we mean that a public key associated with that principal is used for encryption, as in $\{\!|t|\!\}_T^r$, or for signature verification, as in $[\![t]\!]_A$. Message ingredients such as $\mathsf{keytag}, \mathsf{ab\_rq}, \mathsf{ab\_cf}$, etc., are distinctive bit-patterns used to tag data, indicate requests or confirmations, etc. Our notation differs somewhat from Wang's; for instance, his L is our $\mathsf{h}(L)$.

Main exchange. In the first message (Fig. 1), A sends the payload M to B encrypted with a key K, as well as K encrypted with the public encryption key of the trusted third party T. A also sends a digitally signed unit EOO asserting that the payload (etc.) originate with A. The value L serves to identify this session uniquely. In the second message, B countersigns $\mathsf{h}(L), \mathsf{EK}$. In the third message, A discloses K and the random value R used originally to encrypt K for T. B uses this information to obtain M, and also

$$\begin{aligned} A \rightarrow B:&\quad L\,\hat{}\,\mathsf{EM}\,\hat{}\,\mathsf{EK}\,\hat{}\,\mathsf{EOO} \\ B \rightarrow A:&\quad \mathsf{EOR} \\ A \rightarrow B:&\quad K\,\hat{}\,R \end{aligned}$$

$$\text{where:}\quad L = A\,\hat{}\,B\,\hat{}\,T\,\hat{}\,\mathsf{h}(\mathsf{EM})\,\hat{}\,\mathsf{h}(K) \qquad \mathsf{EM} = \{|M|\}_K$$

$$\mathsf{EK} = \{|\,\mathsf{keytag}\,\hat{}\,\mathsf{h}(L)\,\hat{}\,K|\}_T^R \qquad \mathsf{EOO} = [\![\,\mathsf{eootag}\,\hat{}\,\mathsf{h}(L)\,\hat{}\,\mathsf{EK}\,]\!]_A \qquad \mathsf{EOR} = [\![\,\mathsf{eortag}\,\hat{}\,\mathsf{h}(L)\,\hat{}\,\mathsf{EK}\,]\!]_B$$

Figure 1: Wang's protocol: A Successful Run

to reconstruct EK, and thus to validate that the hashes inside EOO are correctly constructed. At the end of a successful exchange, each party deposits the resulting values as a record in its state repository.

Abort and recovery subprotocols. What can go wrong? If the signature keys are uncompromised and the random values K, R are freshly chosen, only two things can fail. Either A fails to receive B's countersigned evidence EOR; or else A receives it, but B fails to receive a correct K, R.

1. If A fails to receive EOR, then A sends the session identifier L and a signed abort request AR to T. T may confirm, and certify the session is aborted, sending a countersigned $[\![\,\mathsf{AR}\,]\!]_T$.
2. If B sends EOR but does not receive K, R, then B asks T to "recover" the session. To do so, B sends $L\,\hat{}\,\mathsf{EK}\,\hat{}\,\mathsf{EOO}\,\hat{}\,\mathsf{EOR}$ to T, inside a signed unit RR indicating that this is a recovery request.

   T can now decrypt the encrypted key $\mathsf{EK} = \{|\,\mathsf{keytag}\,\hat{}\,\mathsf{h}(L)\,\hat{}\,K|\}_T^R$, returning $K\,\hat{}\,R$. If T's attempt to decrypt fails, or yields a values incompatible with the session information, then no harm is done: A will never be able to convince a judge that a valid transaction occurred. Wang's protocol returns an error message that we do not show here [13, Fig. 3].

What should happen if A makes an abort request and B also makes a recovery request, perhaps because EOR was sent but lost in transmission? T services whichever request is received first. When the other party's request is received, T reports the result of that first action. The local behaviors (strands) for A, B in this protocol are shown in Fig. 2. The local sessions (strands) are the paths from a root to a terminal node; there are four paths for A and three paths for B. The solid nodes indicate messages to be sent or received, while the hollow nodes ○ indicate events in which the participants deposit results into their state repositories. This figure is not precise about the forms of the messages, the parameters available to each

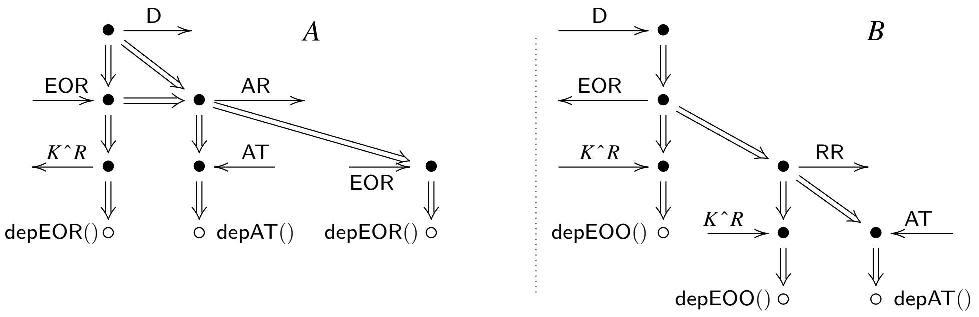

Figure 2: Initiator (A) and Responder (B) Behavior

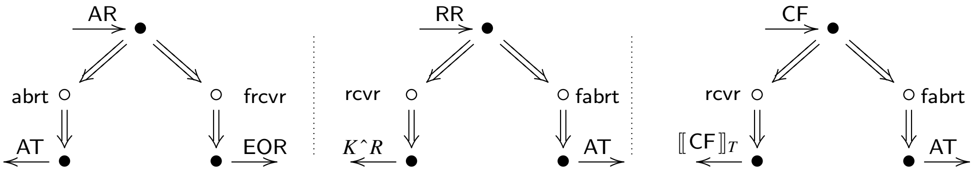

Figure 3: Trusted Third Party: Abort (left), Resolve (center), and Confirm (right) Requests

participant at each point in its run, or the parameters to the state synchronization events. For instance, B does not know whether a claimed EM is really of the form $\{\!|M|\!\}_K$ when first receiving it, nor what M,K would produce the message received. However, the fairness of the protocol is largely independent of these details.

A's abort request AR elicits an abort confirmation $[\![\mathsf{AR}]\!]_T$ if it reaches T first, but it elicits a recovery token $L\hat{\ }\mathsf{EOR}$ if B's recovery request was received first. Likewise, B's recovery request RR elicits $K\hat{\ }R$ if it is received first, but it elicits the abort confirmation $[\![\mathsf{AR}]\!]_T$ if A's abort request was received first. T must synchronize with its state to ensure that these different requests are serviced in compatible ways, depending on whichever arrived first. This compatibility of responses ensures that A,B will execute balanced state changes.

These behaviors of the trusted third party T, together with an additional behavior concerned with dispute resolution, are summarized in Fig. 3. We have indicated here that T's behavior, in response to an abort request AR may lead either to an abort token AT, or else to evidence of receipt EOR. Now, the hollow nodes ○ *guard* the choice of branch. T transmits AR only after a abrt event, and EOR only after a frcvr event. In response to a recovery request RR from B, T may transmit $K\hat{\ }R$ or an abort token AT; however, the former occurs only after a rcvr event and the latter only after a fabrt event. Thus, the essential job for T's long term state in this protocol is to ensure that if an abrt event occurs for session L, then a rcvr never happens for L, and vice versa. This is easily accomplished by a state-based mechanism.

Dispute Resolution. A subtlety in this protocol concerns dispute resolution. Since A receives EOR before disclosing $K\hat{\ }R$, A could choose to abort at this point. A dishonest A could later choose between proving delivery via EOR and proving that this session aborted via the abort token AT. To prevent this, the protocol stipulates that a judge resolving disputes queries B or T for an abort token; it does not accept A's presented EOR if the abort token is also available.

However, this is asymmetric. The abort token is used only by B (or T on B's behalf) to dispute receipt. A can never use it to dispute origin [13, Sec. 4.4], because of essentially the same abuse just mentioned.

For simplicity, we will assume that the judge is identical with T. When asked by A to confirm an EOR, T does so if the session has not aborted. When confirming an EOR, T must ensure that the session will never abort in the future, so that an EOR confirmation is handled similarly to a recovery request. If the session has already aborted, then T returns the abort token instead.

This step may make Wang's protocol undesirable in some cases, where T may no longer be available for dispute resolution. It is also why Wang's protocol can use fewer messages than the four that Pfitzmann-Schunter-Waidner proved to be needed in a fair exchange protocol with asynchronous communication [11].

Our Correction to Wang's Protocol. We have adjusted Wang's protocol. When B's recovery request arrives after A's abort request, B receives $[\![\mathsf{AR}]\!]_T$. In the original description, B receives AR itself.

However, then a dishonest B has a strategy to defeat the fairness of the protocol. Namely, after receiving the first message, B does not reply to A, but immediately requests resolution from T, generally receiving $K\hat{\ }R$ from T. When A requests an abort from T, B attempts to read this abort request off of the network. If successful, B has both AR and $K\hat{\ }R$. Hence, it can subsequently choose whether to insist that the message was delivered, using the valid EOO, or whether to repudiate receipt, using the AR.

Whether this attack is possible depends on the nature of the channel between A and T. Under the usual assumption that the channel is resilient in the sense of ensuring delivery, the attack is possible. If the channel offers both resilience and confidentiality, then the attack would be impossible. We have stipulated that B needs the countersigned $[\![\mathsf{AR}]\!]_T$ to make this attack infeasible on the standard assumption of resiliency only.

Authentication Properties of Wang's Protocol. A *strand* is a (linearly ordered) sequence of nodes $n_1 \Rightarrow \ldots \Rightarrow n_j$, each of which represents either:

Transmission of some message $\mathsf{msg}(n_i)$;

Reception of some message $\mathsf{msg}(n_i)$; or

State synchronization labeled by some *fact*, i.e. a variable-free atomic formula, $E(a_1,\ldots,a_k)$.

A strand may represent the behavior of a principal in a single local session of a protocol, in which case it is a *regular* strand of that protocol, or it may represent a basic adversary activity. Basic adversary activities include receiving a plaintext and a key and transmitting the result of the encryption, and receiving a ciphertext and its matching decryption key, and transmitting the resulting plaintext. We show transmission and reception nodes by bullets $\bullet$ and state synchronization nodes by hollow circles $\circ$.

A *protocol* Π is a finite set of strands, which are the *roles* of the protocol. A strand s is an *instance* of a role $\rho \in \Pi$, if $s = \rho \cdot \alpha$, i.e. if s results from ρ by applying a substitution α to parameters in ρ.

A *bundle* $\mathscr{B}$ is a finite directed acyclic graph whose vertices are strand nodes, and whose arrows are either strand edges$\Rightarrow$ or communication arrows $\rightarrow$. A bundle satisfies three properties:

1. If $m \rightarrow n$, then m is a transmission node, n is a reception node, and $\mathsf{msg}(m) = \mathsf{msg}(n)$.
2. Every reception node $n \in \mathscr{B}$ has exactly one incoming $\rightarrow$ arrow.
3. If $n \in \mathscr{B}$ and $m \Rightarrow n$, then $m \in \mathscr{B}$.

Bundles model possible protocol executions. Bundles may include both adversary strands and regular strands. For more detail, see the Appendix.

Using this notation, we can state two authentication properties that involve A, B. We omit a proof, which use digital signatures in an extremely routine way, given a precise statement of the protocol.

Lemma 2.1 *1. Suppose $\mathscr{B}$ is a bundle in which B's private signature key is uncompromised, and that, in $\mathscr{B}$, A reaches a node marked* depEOR *on a strand with parameters A,B,T,M,K,R. Then B has executed at least the first two nodes of a responder strand, transmitting* EOR*, on a strand with matching parameters.*

2. Suppose $\mathscr{B}$ is a bundle in which A's private signature key is uncompromised, and that, in $\mathscr{B}$, B reaches a node marked depEOO *or* depAT *on a strand with parameters $A,B,T,\mathsf{EM},\mathsf{EK}$. Then A has executed at least the first node of an initiator strand, transmitting* EOO*, on a strand with matching parameters.*

Two authentication properties involving T are also routine applications of rules for digital signatures.

Lemma 2.2 1. *Suppose $\mathscr{B}$ is a bundle in which A and T's private signature keys are uncompromised, and that, in $\mathscr{B}$, A reaches a node marked* depAT. *Then T has completed a strand transmitting* AT *with matching parameters.*

2. *Suppose $\mathscr{B}$ is a bundle in which A and T's private signature keys are uncompromised. If, in $\mathscr{B}$, B reaches a node marked* depAT*, then:*

   (a) A has reached the second node of an aborting strand, transmitting AR*, on a strand with matching parameters.*

   (b) T has reached node transmitting AT *in response to a recovery query* RR *with matching parameters.*

   If instead B reaches a node marked depEOO *then either A has transmitted $K\hat{}R$, or else T has transmitted $K\hat{}R$.*

Clause (2b) is the part of Lemma 2.2 that would be untrue without our adjustment to Wang's protocol. If B receives only AR, then Clause (2a) holds, but not necessarily Clause (2b). This means that T's state might not reflect the abort.

3 Protocol Behavior and Mutable State

We formalize state change using multiset rewriting [3, 7]. Strands contain special *state synchronization events* that synchronize them with the state of the principal executing the strands, as formalized in Definition 3.5.

3.1 Multiset rewriting to maintain state

We formalize mutable state using MSR. A state is a multiset of ground facts $F(t_1,\ldots,t_i)$, where each $F(t_1,\ldots,t_i)$ is the application of a predicate F to some sequence $t_1,\ldots,t_i$. These arguments are messages, and thus do not contain variables; hence, a state Σ is a multiset of ground facts. We write a vector of messages $t,\ldots,t'$ in the form $\vec{t}$.

A rewrite rule ρ takes the form:

$$D(\vec{t_0}),\ldots,F(\vec{t_1}) \xrightarrow{E(\vec{t_2})} G(\vec{t_3}),\ldots,H(\vec{t_4})$$

where now the arguments $\vec{t_0},\ldots,\vec{t_3}$ are vectors of parametric message terms that may contain variables. When replacing these variables with messages, we obtain ground facts. Unlike [7], we label our transitions with a fact $E(\vec{t_2})$, but we do not require existential quantifiers in the conclusions of rules. We will assume that every variable free in $\vec{t_0},\vec{t_1},\vec{t_3},\vec{t_4}$ is also free in $\vec{t_2}$. Thus, a ground instance of $E(\vec{t_2})$ determines ground instances of all the facts $D(\vec{t_0}),\ldots,F(\vec{t_1}),G(\vec{t_3}),\ldots,H(\vec{t_4})$.

We write $\mathsf{lhs}(\rho)$ for $D(\vec{t_0}),\ldots,F(\vec{t_1})$; we write $\mathsf{rhs}(\rho)$ for $G(\vec{t_3}),\ldots,H(\vec{t_4})$; and $\mathsf{lab}(\rho)$ for $E(\vec{t_2})$.

A rule stipulates that the state can change by consuming instances of the facts in its left-hand side, and producing the corresponding instances of the facts in its right hand side. These sets of facts may overlap, in which case the facts in the overlap are required for the rule to apply, but preserved when it executes. A rewrite rule ρ applies to a state Σ_0 when, for some substitution σ,

$$\Sigma_0 = \Sigma_0', D(\vec{t_0}\cdot\sigma),\ldots,F(\vec{t_1}\cdot\sigma),$$

i.e., Σ_0 is the multiset union of Σ_0' with instances of the premises of ρ under σ. The result of applying ρ to Σ_0, using substitution σ, is

$$\Sigma_0', G(\vec{t_3} \cdot \sigma), \ldots, H(\vec{t_4} \cdot \sigma).$$

Since this is a state, the facts $G(\vec{t_3} \cdot \sigma), \ldots, H(\vec{t_4} \cdot \sigma)$ must again be ground; i.e. σ must associate the variables of $\vec{t_3}, \ldots, \vec{t_4}$ with variable-free messages. There may be variables in $\vec{t_3}, \ldots, \vec{t_4}$ that do not occur in $\vec{t_0}, \ldots, \vec{t_1}$. These variables take values nondeterministically, from the point of view of the prior state. In an execution, they may be determined by protocol activities synchronized with the state. Our assumption about the variables in $E(\vec{t_2})$ ensures each ground instance of $E(\vec{t_2})$ determines a σ under which $\vec{t_3}, \ldots, \vec{t_4}$ become ground, and t_2 so to speak summarizes all choices of values for variables.

Definition 3.1 *Let* $\rho = D(\vec{t_0}), \ldots, F(\vec{t_1}) \xrightarrow{E(\vec{t_2})} G(\vec{t_3}), \ldots, H(\vec{t_4})$.

$\Sigma_0 \xrightarrow{\rho,\sigma} \Sigma_1$ *a* ρ, σ *transition from* Σ_0 *to* Σ_1 *iff* Σ_0, Σ_1 *are ground, and there exists a* Σ_0' *such that* $\Sigma_0 = \Sigma_0', D(\vec{t_0} \cdot \sigma), \ldots, F(\vec{t_1} \cdot \sigma)$ *and* $\Sigma_1 = \Sigma_0', G(\vec{t_3} \cdot \sigma), \ldots, H(\vec{t_4} \cdot \sigma)$.

A computation $\mathscr{C}$ *is finite path through states via transitions; i.e.* $\mathscr{C} = \Sigma_0 \xrightarrow{\rho_0,\sigma_0} \Sigma_1 \xrightarrow{\rho_1,\sigma_1} \ldots \xrightarrow{\rho_j,\sigma_j} \Sigma_{j+1}$. $\mathscr{C}$ *is* over *a set of rules R if each* $\rho_i \in R$. *When no ambiguity results, we will also write* $\mathscr{C}$ *in the form:*

$$\mathscr{C} = \Sigma_0 \xrightarrow{E_0(\vec{t_0} \cdot \sigma_0)} \Sigma_1 \xrightarrow{E_1(\vec{t_1} \cdot \sigma_1)} \ldots \xrightarrow{E_j(\vec{t_j} \cdot \sigma_j)} \Sigma_{j+1}.$$

We write $\mathsf{first}(\mathscr{C})$ *for* Σ_0 *and* $\mathsf{last}(\mathscr{C})$ *for* Σ_{j+1}.

In this lemma, we interpret $\setminus, \cup, \subseteq$ as the multiset difference, union, and subset operators.

Lemma 3.2 *Suppose* $(\mathsf{lhs}(\rho_1) \cdot \sigma_1) \cup (\mathsf{lhs}(\rho_2) \cdot \sigma_2) \subseteq \Sigma_0$. *If* $\Sigma_0 \xrightarrow{\rho_1,\sigma_1} \Sigma_1 \xrightarrow{\rho_2,\sigma_2} \Sigma_2$, *then*

$$\exists \Sigma_1' \,.\, \Sigma_0 \xrightarrow{\rho_2,\sigma_2} \Sigma_1' \xrightarrow{\rho_1,\sigma_1} \Sigma_2.$$

Proof. $\Sigma_1 = (\Sigma_0 \setminus (\mathsf{lhs}(\rho_1) \cdot \sigma_1)) \cup (\mathsf{rhs}(\rho_1) \cdot \sigma_1)$, and $\Sigma_2 = (\Sigma_1 \setminus (\mathsf{lhs}(\rho_2) \cdot \sigma_2)) \cup (\mathsf{rhs}(\rho_2) \cdot \sigma_2)$. We define $\Sigma_1' = (\Sigma_0 \setminus (\mathsf{lhs}(\rho_2) \cdot \sigma_2)) \cup (\mathsf{rhs}(\rho_2) \cdot \sigma_2)$. By the assumption, $\Sigma_2 = (\Sigma_1' \setminus (\mathsf{lhs}(\rho_1) \cdot \sigma_1)) \cup (\mathsf{rhs}(\rho_1) \cdot \sigma_1)$. □

3.2 Locality to principals

In our manner of using MSR, all manipulation of state is local to a particular principal, and coordination among different principals occurs only through protocol behavior represented on strands.

Definition 3.3 *A set of rewrite rules R is* localized to principals, *if, for a single distinguished variable p, for every rule* $\rho \in R$, *for each fact* $F(\vec{t})$ *occurring in* ρ *as a premise or conclusion,* $F(\vec{t})$ *is of the form* $F(p, \vec{t'})$.

The principal of *a transition* $\Sigma_0 \xrightarrow{\rho,\sigma} \Sigma_1$ *is* $p \cdot \sigma$.

Thus, only the principal of a transition $\Sigma_0 \xrightarrow{\rho,\sigma} \Sigma_1$ is affected by it. Transitions with different principals are always concurrent. If $p \cdot \sigma_1 \neq p \cdot \sigma_2$ and $(\rho_1, \sigma_1), (\rho_2, \sigma_2)$ can happen, so can the reverse, with the same effect:

Corollary 3.4 *Let R be localized to principals, with* $\rho_1, \rho_2 \in R$, *and* $p \cdot \sigma_1 \neq p \cdot \sigma_2$. *If* $\Sigma_0 \xrightarrow{\rho_1,\sigma_1} \Sigma_1 \xrightarrow{\rho_2,\sigma_2} \Sigma_2$, *then* $\Sigma_0 \xrightarrow{\rho_2,\sigma_2} \Sigma_1' \xrightarrow{\rho_1,\sigma_1} \Sigma_2$, *for some* Σ_1'.

Proof. Since $p \cdot \sigma_1 \neq p \cdot \sigma_2$, the facts on the right hand side of $\rho_1 \cdot \sigma_1$ are disjoint from those on the left hand side of $\rho_2 \cdot \sigma_2$. Hence, ρ_2, σ_2 being enabled in Σ_1, it must also be enabled in Σ_0. Hence, $(\mathsf{lhs}(\rho_1) \cdot \sigma_1) \cup (\mathsf{lhs}(\rho_2) \cdot \sigma_2) \subset \Sigma_0$, and we may apply Lemma 3.2. □

The following definition connects bundles with computations.

Definition 3.5 *Let R be localized to principals.*

1. *An* eventful protocol Π *is a finite set of roles containing nodes of three kinds:*
   (a) *transmission nodes $+t$, where t is a message;*
   (b) *reception nodes $-t$, where t is a message; and*
   (c) *state synchronization events $E_i(p,\vec{t})$.*

   We require that if $E_i(p,\vec{t})$ and $E_j(p',\vec{t'})$ lie on the same strand, then $p = p'$. If a strand s contains a state synchronization $E_i(p,\vec{t})$, then p is the principal of *s.*

2. *Suppose that $\mathscr{B}$ is a bundle over the eventful protocol Π; $\mathscr{C}$ is a finite computation for the rules R; and ϕ is a bijection between state synchronization nodes of $\mathscr{B}$ and transitions $E_i(\vec{t_i})$ of $\mathscr{C}$. $\mathscr{B}$ is* compatible with $\mathscr{C}$ under ϕ *iff*
   (a) *The event $E_i(p,\vec{t})$ at n is the label on $\phi(n)$, and*
   (b) *$n_0 \preceq_{\mathscr{B}} n_1$ implies $\phi(n_0)$ precedes $\phi(n_1)$ in $\mathscr{C}$.*

3. *An* execution *of* Π constrained by *R is a triple $(\mathscr{B},\mathscr{C},\phi)$ where $\mathscr{B}$ is compatible with $\mathscr{C}$ under ϕ.*

If $(\mathscr{B},\mathscr{C},\phi)$ is an execution, then it represents possible protocol behavior $\mathscr{B}$ for Π, where state-sensitive steps are constrained by the state maintained in $\mathscr{C}$. Moreover, the state $\mathscr{C}$ evolves as driven by state synchronizations occurring in strands appearing in $\mathscr{B}$. The bijection ϕ makes explicit the correlation between events in the protocol runs of $\mathscr{B}$ and transitions occurring in $\mathscr{C}$.

3.3 States and Rules for Wang's Protocol

Trusted Third Party State. Conceptually, the trusted third party T_0 maintains a status record for each possible transaction it could be asked to abort or recover. Since each transaction is determined by a label $\mathscr{L}_m(hm,hk) = A\hat{\ }B\hat{\ }T\hat{\ }hm\hat{\ }hk$, where $T = T_0$, it maintains a fact for each such value. This fact indicates either (1) that the no message has as yet been received in connection with this session; or (2) that the session has been recovered, in which case the evidence of receipt is also kept in the record; or (3) that the session has been aborted, in which case the signed abort request is also kept in the record. Thus, the state record for the session with label $\ell = \mathscr{L}_m(hm,hk)$ is a fact of one of the three forms:

$$\mathsf{unseen}(T,\ell) \qquad \mathsf{recovered}(T,\ell,\mathsf{EOR}) \qquad \mathsf{aborted}(T,\ell,\mathsf{AT})$$

Naturally, a programmer will maintain a sparse representation of this state, in which only the last two forms are actually stored. A query for ℓ that retrieves nothing indicates that the session ℓ is as yet unseen.

Four types of events synchronize with T's state. The event $\mathsf{rcvr}(\ell,e)$ deposits a $\mathsf{recovered}(\ell,e)$ fact into the state, and requires the state to contain either an $\mathsf{unseen}(\ell)$ fact or a preexisting $\mathsf{recovered}(\ell,e)$ fact with the same e, which are consumed.

$$\mathsf{unseen}(T,\ell) \xrightarrow{\mathsf{rcvr}(T,\ell,e)} \mathsf{recovered}(T,\ell,e)$$
$$\mathsf{recovered}(T,\ell,e) \xrightarrow{\mathsf{rcvr}(T,\ell,e)} \mathsf{recovered}(T,\ell,e)$$

The second of these forms ensures that repeated rcvr events succeed, with no further state change.

The event $\mathsf{abrt}(T,\ell,a)$ deposits a $\mathsf{aborted}(T,\ell,a)$ fact into the state, and requires the state to contain either an $\mathsf{unseen}(T,\ell)$ fact or a preexisting $\mathsf{aborted}(T,\ell,a)$ fact, which are consumed.

$$\mathsf{unseen}(T,\ell) \xrightarrow{\mathsf{abrt}(T,\ell,a)} \mathsf{aborted}(T,\ell,a)$$
$$\mathsf{aborted}(T,\ell,a) \xrightarrow{\mathsf{abrt}(T,\ell,a)} \mathsf{aborted}(T,\ell,a)$$

Finally, there is an event for a forced recover $\mathsf{frcvr}(T,\ell,e)$ and one for a forced abort $\mathsf{fabrt}(T,\ell,a)$. These may occur when the recovered fact [or respectively, the aborted fact] is already present, so that attempt to abort [or respectively, to recover] must yield the opposite result.

$$\mathsf{recovered}(T,\ell,e) \xrightarrow{\mathsf{frcvr}(T,\ell,e)} \mathsf{recovered}(T,\ell,e)$$
$$\mathsf{aborted}(T,\ell,a) \xrightarrow{\mathsf{fabrt}(T,\ell,a)} \mathsf{aborted}(T,\ell,a)$$

Definition 3.6 *A* GW initial state *is a multiset* Σ *such that:*

1. *No fact* $\mathsf{recovered}(T,\ell,e)$ *or* $\mathsf{aborted}(T,\ell,a)$ *is present in* Σ*;*
2. *For all* ℓ*, the multiplicity* $|\mathsf{unseen}(T,\ell)|_\Sigma$ *of* $\mathsf{unseen}(T,\ell)$ *in* Σ *is at most 1.*

$\mathscr{C}$ *is a* GW computation *if it is a computation using the set* R_W *of the six rules above, starting from a* GW *initial state* Σ_0.

There are several obvious consequences of the definitions. The first says that the multiplicity of facts for a single session ℓ does not increase, and initially starts at 0 or 1, concentrated in $\mathsf{unseen}(T,\ell)$. The next two say that a $\mathsf{recovered}(T,\ell,e)$ fact arises only after a $\mathsf{rcvr}(T,\ell,e)$ event, and a $\mathsf{aborted}(T,\ell,a)$ fact after an $\mathsf{abrt}(T,\ell,e)$ event. Then we point out that a $\mathsf{rcvr}(T,\ell,e)$ event and an $\mathsf{abrt}(T,\ell,a)$ event never occur in the same computation, and finally that a $\mathsf{rcvr}(T,\ell,e)$ event must precede a $\mathsf{frcvr}(T,\ell,e)$ event, and likewise for aborts and forced aborts.

Lemma 3.7 *Let* $\mathscr{C} = \Sigma_0 \overset{\rho_0,\sigma_0}{\longrightarrow} \Sigma_1 \overset{\rho_1,\sigma_1}{\longrightarrow} \dots \overset{\rho_j,\sigma_j}{\longrightarrow} \Sigma_{j+1}$ *be a* GW *computation.*

1. *For any* ℓ *and* $i \leq j+1$*, the sum over all* e,a *of the multiplicities of all facts* $\mathsf{unseen}(T,\ell)$*,* $\mathsf{recovered}(T,\ell,e)$*,* $\mathsf{aborted}(T,\ell,a)$ *is unchanged:*

$$\begin{aligned} 1 \geq |\mathsf{unseen}(T,\ell)|_{\Sigma_0} &= \sum_{a,e}\big(|\mathsf{unseen}(T,\ell)|_{\Sigma_i} + |\mathsf{recovered}(T,\ell,e)|_{\Sigma_i} \\ &+ \; |\mathsf{aborted}(T,\ell,a)|_{\Sigma_i}\big). \end{aligned}$$

2. $|\mathsf{recovered}(T,\ell,e)|_{\Sigma_i} = 1$ *iff* $\exists k < i$, $\mathsf{lab}(\rho_k)\cdot\sigma_k = \mathsf{rcvr}(T,\ell,e)$.
3. $|\mathsf{aborted}(T,\ell,a)|_{\Sigma_i} = 1$ *iff* $\exists k < i$, $\mathsf{lab}(\rho_k)\cdot\sigma_k = \mathsf{abrt}(T,\ell,e)$.
4. *If* $\exists i$, $\mathsf{lab}(\rho_i)\cdot\sigma_i = \mathsf{rcvr}(T,\ell,e)$*, then* $\forall k,a$, $\mathsf{lab}(\rho_k)\cdot\sigma_k \neq \mathsf{abrt}(T,\ell,a)$.
5. *If* $\exists i$, $\mathsf{lab}(\rho_i)\cdot\sigma_i = \mathsf{frcvr}(T,\ell,e)$*, then* $\exists k < i$, $\mathsf{lab}(\rho_k)\cdot\sigma_k = \mathsf{rcvr}(T,\ell,e)$.
6. *If* $\exists i$, $\mathsf{lab}(\rho_i)\cdot\sigma_i = \mathsf{fabrt}(T,\ell,a)$*, then* $\exists k < i$, $\mathsf{lab}(\rho_k)\cdot\sigma_k = \mathsf{abrt}(T,\ell,a)$.
7. *If* $\mathsf{unseen}(T,\ell) \in \Sigma_0$*, then every session* ℓ *request to* T *in Fig. 3 can proceed on some branch.*

Initiator and Responder State. The initiator and responder have rules with empty precondition, that simply deposit records values into their state. These records are of the forms $\mathsf{eor}(A,\ell,\mathsf{EOR},M,K,R)$, $\mathsf{eoo}(B,\ell,\mathsf{EOO},M,K,R)$, and $\mathsf{aborted}(P,\ell,[\![\,[\![\mathsf{ab\_rq}\,\hat{}\,\mathsf{h}(\ell)]\!]_A\,]\!]_T)$. The last is used both by the initiator and the responder. The rules are:

$$\cdot \qquad \xrightarrow{\mathsf{depEOR}(A,\ell,e,M,K,R)} \mathsf{eor}(A,\ell,e,M,K,R)$$

$$\cdot \qquad \xrightarrow{\mathsf{depEOO}(B,\ell,e,M,K,R)} \mathsf{eoo}(B,\ell,e,M,K,R)$$

$$\cdot \qquad \xrightarrow{\mathsf{depAT}(P,\ell,a)} \mathsf{aborted}(P,\ell,a)$$

4 Progress Assumptions

We introduce two kinds of progress properties for protocols. One of them (Def. 4.1) formalizes the idea that certain messages, if sent, must be delivered to a regular participant, i.e. that these messages traverse resilient channels. The second is the idea that principals, at particular nodes in a strand, must progress. We will stipulate that a principal whose next step is a state event, and the current state satisfies the right hand side of the associated rule, then the principal will always take either that step or another enabled step. It is formalized in Def. 4.3.

Definition 4.1 *Suppose that Π is a protocol, and G is a set of nodes $s \downarrow i$ such that for all $s \downarrow i \in G$, s is a role of Π and $s \downarrow i$ is a transmission node. Then G is a set of* guaranteed delivery assumptions *for Π.*

A transmission node n on a strand s' is a guaranteed delivery node *for Π, G if it is an instance $n = (s \downarrow i) \cdot \alpha$ of the i^{th} node of some role $s \in \Pi$, and $s \downarrow i \in G$.*

Let $\mathscr{B}$ be a bundle for Π. $\mathscr{B}$ satisfies guaranteed delivery *for G if, for every guaranteed delivery node $n \in \mathscr{B}$, there is a unique node $m \in \mathscr{B}$ such that $n \rightarrow_{\mathscr{B}} m$, and moreover m is regular.*

There are three ingredients here. First, n's transmission should be received somewhere. Second, it should be received at most once. Finally, the recipient should be regular. For our progress condition, however, we want a stronger condition than this guaranteed delivery property. In particular, we also want to stipulate that if a guaranteed transmission node can be added, and its message can be delivered, then it will be added together with one matching reception node. However, for this we need to define the right notion of "can." Thus, we define the unresolved nodes of a bundle, using $n \sim m$, which means that n and m are similar in the following sense:

Definition 4.2 *Regular nodes n', m' are* similar, *written $n' \sim m'$, if the initial segments of the strands they lie on, $n \Rightarrow \ldots \Rightarrow n'$ and $m \Rightarrow \ldots \Rightarrow m'$, (1) are of the same length; (2) corresponding nodes have the same direction (transmission, reception, or state synchronization); and (3) corresponding nodes have the same message or state synchronization event label.*

A regular node n_0 is unresolved *in $\mathscr{B}$ if $n_0 \Rightarrow n_1$ and for some $n_0' \in \mathscr{B}$, $n_0' \sim n_0$ but for all $n_1' \in \mathscr{B}$, $n_0' \not\Rightarrow n_1'$.*

A node n_0 is unresolved if it *can* progress to some n_1, but a similar $n_0' \in \mathscr{B}$ *has not* progressed. Thus, substituting a similar node for a node in $\mathscr{B}$, we obtain a bundle $\mathscr{B}'$ to which this transition may be added.

Definition 4.3 *Let $\mathscr{E} = (\mathscr{B}, \mathscr{C}, \phi)$ be an execution of Π, G constrained by R. $\mathscr{E}$ is a* stable execution *if (1) $\mathscr{B}$ satisfies guaranteed delivery for G; (2) there are no enabled transmission edges for $\mathscr{B}$; and (3) there are no enabled state edges for $\mathscr{E}$, where we define* enabled transmission and state edges *as follows:*

1. *$n_0 \Rightarrow n_1$ is an* enabled transmission edge for $\mathscr{E}$ *if:*
   (a) *n_0 is unresolved in $\mathscr{B}$;*
   (b) *n_1 is a guaranteed delivery node; and*
   (c) *there is a regular reception node m_1 with $\mathsf{msg}(m_1) = \mathsf{msg}(n_1)$ where either*
      i. *m_1 is the first node on its strand, or else*
      ii. *$m_0 \Rightarrow m_1$, where m_0 is unresolved in $\mathscr{B}$.*
2. *$n_0 \Rightarrow n_1$ is an* enabled state edge for $\mathscr{E}$ *if:*
   (a) *n_0 is unresolved in $\mathscr{B}$;*
   (b) *n_1 is a state synchronization node with event $E(p,\vec{t})$; and*
   (c) *$\exists \rho \in R$ and σ s.t. $\mathsf{lab}(\rho)\cdot\sigma = E(p,\vec{t})$ and $\mathsf{lhs}(\rho)\cdot\sigma \subseteq \mathsf{last}(\mathscr{C})$.*

In a stable execution, each strand has reached a "stopping point," where no transmission with guaranteed delivery (and matching reception) is waiting to happen, and no state synchronization event is waiting to happen. A protocol Π and rules R drive the evolution of state through states satisfying some balance property Ψ means that when $\mathscr{E} = (\mathscr{B}, \mathscr{C}, \phi)$ is a stable execution for Π, R, and $\Psi(\mathsf{first}(\mathscr{C}))$, then $\Psi(\mathsf{last}(\mathscr{C}))$.

Guaranteed Delivery for Wang's Protocol. The guaranteed delivery assumptions for Wang's protocol are not surprising. They are the messages transmitted on resilient channels between the principals and the Trusted Third Party. These are A's transmission of AR and B's transmission of RR in Fig 2, and T's six transmissions in Fig. 3.

Progress in Wang's Protocol. No protocol can protect principals that do not follow it. Thus, correctness conditions are stated for stable executions in which at least one of A, B comply with the protocol. We also assume that the trusted third party T merits trust, and also complies with the protocol. A principal P is *compliant* in a bundle $\mathscr{B}$ if $P \in \{A, B\}$ and P's signing key is used only in accordance with Π_{GW} in $\mathscr{B}$; or if $P = T$, the trusted third party, and T's signing and decryption keys are used only in accordance with Π_{GW} in $\mathscr{B}$.

Henceforth, let $\mathscr{E} = (\mathscr{B}, \mathscr{C}, \phi)$ be a GW-execution. Let Σ_0 and Σ_j be the first and last states of $\mathscr{C}$. For each label ℓ occurring in an A initiator strand or a B responder strand in $\mathscr{B}$, assume that $\mathsf{unseen}(T, \ell) \in \Sigma_0$.

Lemma 4.4 (GW **Progress**) *Let S be a set of principals compliant in $\mathscr{E}$, with $T \in S$. There exists a stable $\mathscr{E}' = (\mathscr{B}', \mathscr{C}', \phi')$, such that (1) $\mathscr{E}'$ extends $\mathscr{E}$; (2) the principals S are compliant in $\mathscr{E}'$; and (3) $p = T$ if p is the principal of any regular strand of $\mathscr{B}'$ that does not appear in $\mathscr{B}$.*

If s is an initiator or TTP strand with $\mathscr{B}'$-height ≥ 1, then its $\mathscr{B}'$-height is its full length. If s is a responder strand with $\mathscr{B}'$-height ≥ 2, then its $\mathscr{B}'$-height is its full length.

Proof. Inspecting Fig. 2, we see that an initiator strand of $\mathscr{B}$-height 1 may progress by sending a guaranteed-delivery AR, which is also possible for an initiator strand that has received EOR. The guaranteed delivery rule requires the first node of some T strand receiving AR. By Lemma 3.7, Clause 7, some T state synchronization event is enabled, after which T makes a guaranteed-delivery transmission. Thus, A receives AT or EOR. Since its deposit state synchronization events have empty precondition, A will complete its strand. The analysis for responder strands is similar. □

That is, we may regard starting a strand in $\mathscr{B}$, or—for a responder—sending its EOR message, as a promise to progress regularly in the future, as required by Def. 4.3. Moreover, new strands that begin in $\mathscr{B}'$, not $\mathscr{B}$, belong only to the TTP T. In $\mathscr{B}'$, these strands have terminated by reaching its full length.

5 Correctness of Wang's protocol

We now summarize our conclusions in a theorem that puts together the different elements we have discussed.

Theorem 5.1 *Let* $\mathscr{E} = (\mathscr{B}, \mathscr{C}, \phi)$ *be a* stable GW*-execution with* $\mathsf{unseen}(T, \ell) \in \Sigma_0$.

1. *If* $\mathsf{eoo}(B, \ell, \mathsf{EOO}, M, K, R) \in \Sigma_j$ *but* $\notin \Sigma_0$*, then for compliant* A, $\mathsf{eor}(A, \ell, \mathsf{EOR}, M, K, R) \in \Sigma_j$.
2. *If* $\mathsf{eor}(A, \ell, \mathsf{EOR}, M, K, R) \in \Sigma_j$ *but* $\notin \Sigma_0$*, then for compliant* B*, either* $\mathsf{eoo}(B, \ell, \mathsf{EOO}, M, K, R) \in \Sigma_j$ *or else* $\mathsf{aborted}(B, \ell, \mathsf{AT}) \in \Sigma_j$.

Proof. 1. By the state rules for B, $\mathsf{depEOO}(B, \ell, e, M, K, R)$ has occurred in $\mathscr{C}$. Hence, B has reached one of the two $\mathsf{depEOO}()$ nodes shown in Fig 2, with parameters B, ℓ, e, M, K, R. Hence, by Lemma 2.1, Clause 2, A has executed at least the first node of an initiator strand, transmitting EOO, on a strand with matching parameters. Since $\mathscr{E}$ is stable, by Thm. 4.4, A's strand has full height. Thus, either $\mathsf{depEOR}()$ or $\mathsf{depAT}()$ has occurred with matching parameters.

However, if $\mathsf{depAT}()$ has occurred at A, then A does not transmit $K\hat{}R$. Moreover, since A has received AT, T has transmitted AT by Lemma 2.2, Clause 1. Hence, by Lemma 3.7, Clause 4, $\mathscr{C}$ does not contain a $\mathsf{rcvr}(T, \ell, e)$ event. Thus, contrary to Lemma 2.2, Clause 2, T has not transmitted $K\hat{}R$. Hence, $\mathsf{depEOR}()$ has occurred.

2. By the state rules for A, $\mathsf{depEOR}(A, \ell, e, M, K, R)$ has occurred in $\mathscr{C}$. Hence, A has reached one of the two $\mathsf{depEOR}()$ nodes shown in Fig 2, with parameters A, ℓ, e, M, K, R. Hence, by Lemma 2.1, Clause 1, B has executed at least the first two nodes of a responder strand, transmitting EOR, on a strand with matching parameters. Since $\mathscr{E}$ is stable, by Thm. 4.4, B's strand has full height. Thus, either $\mathsf{depEOO}()$ or $\mathsf{depAT}()$ has occurred at B with matching parameters. □

Conclusion. This formalism has also been found to be convenient to model the interface to a cryptographic device, the Trusted Platform Module, which combines cryptographic operations with a repository of state. Thus, it appears to be a widely applicable approach to the problem of combining reasoning about cryptographic protocols with reasoning about state and histories.

References

[1] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE J. Sel. Areas in Comms.*, 18(4):593–610, 2000.

[2] Jan Cederquist, Mohammad Torabi Dashti, and Sjouke Mauw. A certified email protocol using key chains. In *Advanced Information Networking and Applications Workshops/Symposia (AINA'07), Symposium on Security in Networks and Distributed Systems (SSNDS07)*, volume 1, pages 525–530. IEEE CS Press, 2007.

[3] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proceedings, 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[4] Rohit Chadha, John C. Mitchell, Andre Scedrov, and Vitaly Shmatikov. Contract signing, optimism, and advantage. In *Concur — Concurrency Theory*, LNCS, pages 366–382. Springer, 2003.

[5] Mohammad Torabi Dashti. *Keeping Fairness Alive*. PhD thesis, Vrije Universiteit, Amsterdam, 2007.

[6] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Searching for shapes in cryptographic protocols. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, number 4424 in LNCS, pages 523–538. Springer, March 2007. Extended version at URL:`http://eprint.iacr.org/2006/435`.

[7] Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004. Initial version appeared in *Workshop on Formal Methods and Security Protocols*, 1999.

[8] Shimon Even and Yacov Yacobi. Relations among public key signature systems. Technical Report 175, Computer Science Departament, Technion, 1980.

[9] Joshua D. Guttman. Cryptographic protocol composition via the authentication tests. In Luca de Alfaro, editor, *Foundations of Software Science and Computation Structures (FOSSACS)*, number 5504 in LNCS, pages 303–317. Springer, March 2009.

[10] Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002. Conference version appeared in *IEEE Symposium on Security and Privacy*, May 2000.

[11] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Optimal efficiency of optimistic contract signing. In *Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 113–122, New York, May 1998. ACM.

[12] Michael Rabin. How to exchange secrets by oblivious transfer. Technical report, Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981. Available at http://eprint.iacr.org/2005/187.

[13] Guilin Wang. Generic non-repudiation protocols supporting transparent off-line TTP. *Journal of Computer Security*, 14(5):441–467, 2006.

A Messages and Protocols

In this appendix, we provide an overview of the current strand space framework; this section is essentially identical with part of [9].

Message Algebra. Let A_0 be an algebra equipped with some operators and a set of homomorphisms $\eta\colon \mathsf{A}_0 \to \mathsf{A}_0$. We call members of A_0 *atoms*.

For the sake of definiteness, we will assume here that A_0 is the disjoint union of infinite sets of *nonces*, *atomic keys*, *names*, and *texts*. The operator $\mathsf{sk}(a)$ maps names to (atomic) signature keys, and K^{-1} maps an asymmetric atomic key to its inverse, and a symmetric atomic key to itself. Homomorphisms η are maps that respect sorts, and act homomorphically on $\mathsf{sk}(a)$ and K^{-1}.

Let X is an infinite set disjoint from A_0; its members—called *indeterminates*—act like unsorted variables. A is freely generated from $\mathsf{A}_0 \cup X$ by two operations: encryption $\{|t_0|\}_{t_1}$ and tagged concatenation $\mathit{tag}\ t_0\hat{\ }t_1$, where the tags tag are drawn from some set TAG. For a distinguished tag nil, we write $\mathit{nil}\ t_0\hat{\ }t_1$ as $t_0\hat{\ }t_1$ with no tag. In $\{|t_0|\}_{t_1}$, a non-atomic key t_1 is a symmetric key. Members of A are called *messages*.

A homomorphism $\alpha = (\eta,\chi)\colon \mathsf{A} \to \mathsf{A}$ consists of a homomorphism η on atoms and a function $\chi\colon X \to \mathsf{A}$. It is defined for all $t \in \mathsf{A}$ by the conditions:

$$\begin{array}{llcl} a\cdot\alpha = \eta(a), & \text{if } a \in \mathsf{A}_0 & & \{|t_0|\}_{t_1}\cdot\alpha = \{|t_0\cdot\alpha|\}_{t_1\cdot\alpha} \\ x\cdot\alpha = \chi(x), & \text{if } x \in X & & \mathit{tag}\ t_0\hat{\ }t_1\cdot\alpha = \mathit{tag}\ t_0\cdot\alpha\hat{\ }t_1\cdot\alpha \end{array}$$

Thus, atoms serve as typed variables, replaceable only by other values of the same sort, while indeterminates x are untyped. Indeterminates x serve as blank slots, to be filled by any $\chi(x) \in \mathsf{A}$. Indeterminates and atoms are jointly *parameters*.

Messages are abstract syntax trees in the usual way:

1. Let ℓ and r be the partial functions such that for $t = \{|t_1|\}_{t_2}$ or $t = \mathit{tag}\ t_1\hat{\ }t_2$, $\ell(t) = t_1$ and $r(t) = t_2$; and for $t \in \mathsf{A}_0$, ℓ and r are undefined.

2. A *path* p is a sequence in $\{\ell, r\}^*$. We regard p as a partial function, where $\langle\rangle = \mathsf{Id}$ and $\mathsf{cons}(f,p) = p \circ f$. When the rhs is defined, we have: 1. $\langle\rangle(t) = t$; 2. $\mathsf{cons}(\ell,p)(t) = p(\ell(t))$; and 3. $\mathsf{cons}(r,p)(t) = p(r(t))$.

3. *p traverses a key edge* in t if $p_1(t)$ is an encryption, where $p = p_1{}^\frown\langle r\rangle{}^\frown p_2$.

4. *p traverses a member of S* if $p_1(t) \in S$, where $p = p_1 {}^\frown p_2$ and $p_2 \neq \langle\rangle$.
5. *t_0 is an ingredient of t*, written $t_0 \sqsubseteq t$, if $t_0 = p(t)$ for some p that does not traverse a key edge in t.
6. *t_0 appears in t*, written $t_0 \ll t$, if $t_0 = p(t)$ for some p.

A single local session of a protocol at a single principal is a *strand*, containing a linearly ordered sequence of transmissions, receptions, and state synchronization events that we call *nodes*. In Figs. 2–3, the columns of nodes connected by double arrows $\Rightarrow$ are strands.

Assumption 1 *Strands and nodes are disjoint from* A.

A message t_0 *originates* at a node n_1 if (1) n_1 is a transmission node; (2) $t_0 \sqsubseteq \mathsf{msg}(n_1)$; and (3) whenever $n_0 \Rightarrow^+ n_1$, $t_0 \not\sqsubseteq \mathsf{msg}(n_0)$.

Thus, t_0 originates when it was transmitted without having been either received, transmitted, or synchronized previously on the same strand. Values assumed to originate only on one node in an execution—*uniquely originating* values—formalize the idea of freshly chosen, unguessable values. Values assumed to originate nowhere may be used to encrypt or decrypt, but are never sent as message ingredients. They are called *non-originating* values. For a non-originating value K, $K \not\sqsubseteq t$ for any transmitted message t. However, $K \ll \{|t_0|\}_K \sqsubseteq t$ possibly, which is why we distinguish $\sqsubseteq$ from $\ll$. See [10, 6] for more details.

Protocols. A *protocol* Π is a finite set of strands, representing the roles of the protocol. Their instances result by replacing A, B, K, M, etc., by any names, symmetric key, text, etc. Each protocol also contains the *listener* role $\mathsf{Lsn}[y]$ with a single reception node in which y is received. The instances of $\mathsf{Lsn}[y]$ are used to document that values are available without cryptographic protection.

Indeterminates represent messages received from protocol peers, or passed down as parameters from higher-level protocols. Thus, we require:

If n_1 is a node on $\rho \in \Pi$, with an indeterminate $x \ll \mathsf{msg}(n_1)$,

then $\exists n_0$, $n_0 \Rightarrow^* n_1$, where n_0 is a reception node and $x \sqsubseteq \mathsf{msg}(n_0)$.

So, an indeterminate is received as an ingredient before appearing in any other way. We say that a strand s is *in $\mathscr{B}$* if s has at least one node in $\mathscr{B}$.

Proposition A.1 *Let $\mathscr{B}$ be a bundle. $\preceq_{\mathscr{B}}$ is a well-founded partial order. Every non-empty set of nodes of $\mathscr{B}$ has $\preceq_{\mathscr{B}}$-minimal members. If $a \sqsubseteq \mathsf{msg}(n)$ for any $n \in \mathscr{B}$, then a originates at some $m \preceq_{\mathscr{B}} n$.*