A FORMAL ANALYSIS OF EXCHANGE OF DIGITAL SIGNATURES

Rohit Chadha

A Dissertation in Mathematics

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2003

Supervisor of Dissertation

Graduate Group Chairperson

Acknowledgments

I owe a special debt of gratitude to my advisor, Andre Scedrov, for his collaboration, insight, guidance and support that made this thesis possible. His faith in my ability to complete this research has been indispensable.

I wish to express my thanks to several colleagues for their collaboration, help and advice. Chapters 2 and 3 are the result of very useful and productive collaboration with Max Kanovich and Andre Scedrov. Chapters 4, 5 and 6 are the result of a very fruitful collaboration with John Mitchell, Andre Scedrov and Vitaly Shmatikov. A special thanks to John Mitchell whose suggestions at various stages have immensely helped the exposition.

I would also like to thank Karthikeyan Bhargavan, Iliano Cervesato, David Dill, Satyaki Dutta, Shimon Even, Peter Freyd, Dieter Gollmann, Joshua Guttman, Carl Gunter, Steve Kremer, Jean-Francois Raskin, Dahlia Malkhi, Catherine Meadows, Jonathan Millen, Sylvan Pinsky and Paul Syverson for interesting and helpful discussions. Their comments have been invaluable in this research.

Thanks to Ted Chinburg, Carl Gunter and Andre Scedrov for serving on my

Ph.D. committee.

A special thanks to the staff of Department of Mathematics at the University of Pennsylvania for ensuring the smooth functioning of various administrative aspects.

Finally, I owe an eternal gratitude to my parents and my many friends. This work would not have possible without their constant encourangement and support.

The author was partially supported by DoD MURI "Semantic Consistency in Information Exchange" as ONR Grant N00014-97-1-0505, NSF grant CCR-9800785, NSF grant CCR-0098096, and by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" as ONR Grant N00014-01-1-0795, Additional support for the author also came in the form of fellowships from the Department of Mathematics at the University of Pennsylvania and from School of Arts and Sciences Dissertation Fellowship.

The results in chapters 2 and 3 were announced in a paper that appeared in the Eight ACM Conference on Computer and Communications Security [15].

Some of the results and definitions in chapters 4 and 5 will be announced in the Fourteenth International Conference on Concurrency Theory [16].

ABSTRACT

A FORMAL ANALYSIS OF EXCHANGE OF DIGITAL SIGNATURES

Rohit Chadha

Advisor: Andre Scedrov

A fair signature exchange protocol lets two parties exchange digital signatures on a specified text. In optimistic protocols, it is possible for two 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. Each signer may have one or more objectives, ranging from "optimistically" trying to put the exchange in place as quickly as possible to maliciously manipulating the other party to gain an advantage. We study in detail the optimistic two-party signature exchange protocol of Garay, Jakobsson and MacKenzie [28] using a game-theoretic framework and show that no signer enjoys an advantage over an honest counterparty. In this setting, we employ the formal inductive proof methods previously used in the formal analysis of simpler, trace-based properties of authentication protocols. We extend this game-theoretic framework to include concepts of preferred behavior and analyze a class of signature exchange protocols. In the process of establishing relationships amongst various protocol properties obtained in the literature, we obtain a fundamental impossibility result: in any fair, optimistic protocol there is a point at which one signer realizes an advantage over an optimistic opponent.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Overview of the problem. We start by giving an informal overview of the problem that we study in this thesis. Later on, we shall detail the technical obstacles that we encountered in our study and our approach in tackling those.

Recent spread of the Internet has highlighted the need of use of cryptographic protocols. These protocols use cryptographic primitives and provide recipes for the participants, ensuring certain desired guarantees of security and trust. The particular problem that we are interested in, is two-party exchange of digital signatures on a pre-specified text. Digital signatures [46] are cryptographic primitives that serve as *universally verifiable* proofs of origin; a digital signature by Alice on a bit string may be used by Bob to convince an *outsider*, *e.g.*, a court, that Alice indeed generated that signature.

A variety of signature-exchange protocols have been proposed in the literature, including gradual-release two-party protocols [5, 7, 18] and fixed-round protocols that rely on an adjudicating "trusted third party" [1, 2, 3, 28, 38, 48]. Some useful properties of signature exchange protocols are *fairness*, which means that either both signing parties, or *signers*, get each others signature or neither does, and *effectiveness,* which means that each signer has some recourse to avoid unbounded waiting. The reason for using a trusted third party in fixed-round protocols is a basic limitation [24, 42] related to the well-known impossibility of distributed consensus in the presence of faults [26]: no fixed-round two-party protocol can be fair. Although there is a trivial trusted third party protocol, in which both signers always send their signatures directly to the third party, some effort is required to produce protocols that are fair, effective, and usefully minimize demands on the third party.

A trusted third party can enforce the exchange after it witnesses a partial completion of the protocol. In *optimistic* protocols [10, 3, 28, 9] the trusted third party is contacted only in case of a dispute, otherwise the protocol can be completed without involving the third party. The reason for designing optimistic protocols is that if a protocol is widely or frequently 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.

Effectiveness and fairness are *trace-based* properties. A trace of a protocol is a sequence of steps that can take place starting from the initial configuration. What other properties should a signature-exchange protocol guarantee? Another kind of symmetry property desirable in distributed signature-exchange was identified in [28]: a fair protocol is said to be *abuse-free* if, at any stage of the protocol, it is impossible for any signer, say A, to be able to prove to an outside challenger that A has the power to choose between *completing* the exchange and *aborting* the exchange. (It is a consequence of fairness that there are two possible outcomes: either the exchange is completed, *i.e.*, both the parties get each other's signatures, or the exchange is aborted, *i.e.*, neither party gets the counterparty's signature).

A family of signature-exchange protocols that illustrates some of the main issues uses four messages (here A and B are the two signers):

$$\begin{array}{rl} A \to B: & \text{commitment to sign} \\ B \to A: & \text{commitment to sign} \\ A \to B: & \text{signature on pre-agreed text} \\ B \to A: & \text{signature on pre-agreed text} \end{array}$$

When A and B complete these four messages, the session ends without requiring a third party. However, if A or B "becomes impatient" at some point, they may progress towards resolution through the adjudicating third party. For example, after A receives a commitment from B, A can send B's commitment, together with A's own signature and receive a verifiable statement enforcing the exchange. Since B can act similarly, each party may become bound through the adjudicator once they commit. However, only A can go to the adjudicator and receive assurance that the adjudicator will not act to enforce the exchange. The restriction that B cannot abort through the third party compensates A for the disadvantage of committing first. Although the adjudicating third party may be needed for progression, the adjudicator could become a communication bottleneck if used consistently in a large number of runs. Assuming the third party operates as a service for many transactions, we can expect some financial cost or potential delay as a result.

We can see the importance of ability to choose an outcome by considering online stock trading with signed documents for each trade. A customer may want a signed document to prevent a broker who does not complete a trade from saying that the request was never received. Similarly, a broker may want a signed document as proof that it is acting as requested. 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. Once the buyer commits to the purchase, he cannot use the committed funds for other purposes. However, if he wishes to avoid the extra cost or potential delay associated with contacting the third party, the buyer is likely to wait sometime for the seller to respond. Since the broker can abort the protocol, 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 sale unprofitable, she can abort against the wishes of the buyer. 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 know exactly or approximately, gives the broker a period where she can decide *unilaterally* whether to abort or complete the exchange.

As we see in the above family of signature exchange protocols, optimistic protocols involve several subprotocols that allow the signature exchange to be completed normally, or aborted or enforced by the adjudicating trusted third party. This makes reasoning about such protocols complicated. Formal methods have been used for study of fair exchange protocols and include FDR [32] and CSP [44]. A finite-state model checker Murφ was used in [45] and in [19] to perform an extensive study of signature exchange protocols. Game theory has also been applied to the study of fair exchange protocols in [15, 35, 36, 12].

In our analysis, we adopt the multiset-rewriting formalism for protocol analysis [14, 13, 22], which reflects the two basic assumptions of the Dolev-Yao model [41, 21, 47], perfect cryptography coupled with nondeterministic computation on the part of the adversary. These assumptions provide an idealized setting in which protocol analysis becomes relatively tractable.

Using multiset-rewriting formalism, we do an extensive study of a revised version of the two-party optimistic signature-exchange protocol of Garay-Jakobsson-Mackenzie protocol [28]. Assuming perfect cryptography and honesty of the trusted third party, we show that for multiple concurrent runs, the revised protocol is fair and effective for honest signers [15]. In our analysis, we assumed that the trusted third party is always honest. The keys to fairness and effectiveness lie in the database properties of the trusted third party and the properties of the communication channels that each party has to the trusted third party.

Even though abuse-freeness was identified in [28], a formal definition was not given. Recall that a fair protocol is said to be abuse-free if, at any stage of the protocol, it is impossible for any signer, say A, to be able to prove to an outside challenger that A has the power to choose between *completing* the exchange and *aborting* the exchange. Hence any approach to formalize abuse-freeness would have to account for two things: a signer's power to unilaterally decide the outcome of the protocol, and its ability to demonstrate this power to an outsider. In [15], we formalized a signer's ability to unilaterally decide the outcome of the protocol if the other signer is honest. As far as we know, this was the first attempt to formalize a signer's power to unilaterally decide the outcome of the protocol.

Using this formalization, we show that the revised protocol is *balanced for honest signers*: no signer can unilaterally decide the outcome if the other signer is honest. In particular, if any one signer is following the protocol, we prove that, at any stage of the protocol, the other signer does not have both the power to complete the exchange as well as the power to abort it. Balance is not a trace-based property and is naturally formulated in terms of game strategies.

A key ingredient in the proof of balance of honest signers in the revised protocol, is the ability of the signers to contact the trusted third party for error recovery. An honest signer can non-deterministically exercise any of the options available to it, including contacting the trusted third party for error recovery. However, as we noted in the discussion of the stock trading example honest signers may have certain preferences. For example, in order to avoid extra cost, an *optimistic signer* would rather wait for its counterparty than rush to the third party for error recovery. Indeed, 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. The other signer may exploit this reluctance to contact the trusted third party. Indeed in the stock trading example discussed earlier, the buyer's reluctance to contact the trusted third party gives an advantage to the broker.

In order to capture the nuisances of the natural behavior of the signers, we revise our previous model. Using the notation of multiset-rewriting formalism to characterize possible protocol actions, we give a model to study a class of two-party signature exchange protocols. We formalize the properties from the literature on fixed-round signature protocols in this model, and establish relationships between them. One basic modeling innovation is an *untimed* nondeterministic setting that provides a semantics for signer preferences. We study two kinds of biased participants: *optimistic* and *interested* signers.

Intuitively, optimistic behavior in signature exchange protocols 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 signer Alice is optimistic, and the other signer Bob chooses to continue the protocol by responding, then Alice waits for Bob's message rather than contact the third party. An interested signer is one who waits for some period of time before contacting the third party to abort the exchange. Please note that while an optimistic Alice waits for Bob before contacting the trusted third party with a request to enforce or abort the exchange, an interested Alice waits for Bob only if the other option is to contact the trusted third party with a request to abort the exchange. An interested Alice will not wait for Bob if she may contact trusted third party with a request to enforce the exchange.

Since the value of an optimistic protocol lies in what it offers to an optimistic signer, we evaluate protocols subject to the assumption that one signer or another follows an optimistic strategy. In evaluating protocol performance for optimistic signers, we prove that in every fair and optimistic protocol, there is point where one of the signers realizes an *advantage* over its optimistic counterparty: at the point of advantage, the signer controls the outcome of the protocol. 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 to whatever degree a protocol allows signers to avoid the third party, the protocol proportionally gives one of the signers unilateral control over the outcome of the protocol.

Since advantage against an optimistic agent cannot be eliminated, the most a protocol can provide an optimistic signer is to prevent the opponent from *proving* that it has the advantage. In this sense, the security guarantees provided by abuse-free protocols [3, 28] are the strongest possible. We are now ready to describe the technical obstacles that we encountered in our study and our approach in tackling those.

Technical obstacles encountered and our approach. We start by an extensive formal analysis of the signature-exchange protocol of Garay, Jakobsson and Mackenzie (GJM) [28]. This analysis is carried out in chapters 2 and 3. A formal treatment of the protocol was first done in [45] using a finite-state model checker, Murφ, resulting in the discovery of an anomaly in the protocol and a proposed fix. We take this revised protocol as our starting point [15].

Why is analysis of this protocol complicated? This is for several reasons. For example, this protocol involves several subprotocols that allow the signature exchange to be completed normally, or aborted or enforced by the adjudicating trusted third party. Furthermore, several pairs of signers may be using the protocol and these instances may interact with each other non-trivially. Some of these signers may also be dishonest and may deviate from the protocol. Also, in order to ensure *effectiveness* and *fairness*, this protocol assumes some special conditions on the communication channels which should be captured by our formalism. Recall that a protocol is said to be effective if it provides means to the signers to prevent unbounded waiting, and a protocol is said to fair if it is the case that either both signers get each other's signatures or none does. Furthermore, this protocol uses some subtle cryptographic primitives in order to ensure abuse-freeness. We discuss each of these issues in detail.

A key decision in our analysis was to choose a formalism that allowed us to model subprotocols, multiple instances of the protocol, cryptographic primitives, the communication channels and dishonest behavior of the signers. We chose multiset-rewriting formalism (MSR) [14, 13, 22] to characterize protocol actions. This formalism allows to us model subprotocols and multiple instances in a natural way.

An important ingredient of the analysis was the modeling of the cryptographic primitives of the protocol. Although the exact definition of these primitives uses probabilistic polynomial-time algorithms, we model a black-box version of these primitives: the mathematical details are abstracted away and the essential properties of these primitives are captured as a set of rewrite rules in our formalism (see section 2.5). Before our work, MSR was used primarily to study simpler authentication protocols that employed cryptographic primitives like encryption [11]. Nevertheless, we were able to express the black-box versions of the cryptographic primitives of the GJM protocol in this formalism.

Another important decision was to model the communication channels in the protocol. In our analysis, we assume that the communication channel between the signers is under the control of a ”Dolev-Yao” [41, 21, 47] *intruder*. An intruder may intercept messages, decompose these messages, compose and send new messages on these channels.

The communication channels to the trusted third party are however special. In [28], these channels are assumed to be private. In our analysis, however, we shall relax this condition and assume that the channel between a signer and the third party are

a) *write-protected*, *i.e.*, nobody other than the signer and the third part can write on these channels, and

b) *transparent*, *i.e.*, this channel never looses this message unless the intended recipient reads it.

The intruder can observe messages on these channels, but not block or delay them.

One of the decisions was how to model dishonest signers. We consider a *strongly dishonest* signer which shares its private keys with the Dolev-Yao intruder prior to the execution of the protocol. On the other hand, we assume that the trusted third party is well-behaved and does not play the role of a party interested in exchanging signatures.

The trusted third party maintains a database of all the protocol instances it has enacted upon. A trusted third party is said to be *accidentally corrupt* [45], if the database of the third party is visible to the intruder. Although we are not modeling dishonest behavior of the trusted third party, our modeling does cover accidentally corrupt behavior of the trusted third party [45]. This is because in our modeling, the intruder may observe all the traffic to/from the trusted third party.

The authors of the GJM protocol did claim *trusted third party accountability* for their protocols. A signature-exchange protocol is said to be third party accountable [28], if whenever an honest signer is cheated because the third party misbehaved, the honest signer is capable of proving to an arbitrator that the third party misbehaved. However we shall not consider this property in our analysis.

Our analysis of the protocol reveals an anomaly in the protocol that was missed in [45]: fairness may be compromised even if we use the proposed fixed protocol in [45]. We propose a fix for this new anomaly, and analyze our revised protocol. For the new protocol, we state and prove that effectiveness and fairness hold for honest signers and for multiple concurrent runs.

The proof of effectiveness is by inductive methods and relies on an honest signer's ability to contact the trusted third party at any stage of the the protocol, and on the properties of the special communication channels between the signers and the trusted third parties.

Using effectiveness, we show that fairness holds for honest signers by inductive methods. The key to fairness lies in the database of the trusted third party. The database enjoys two properties: *persistence*, *i.e.*, an entry once created always remains in the database, and *consistency*, *i.e.*, the database does not have conflicting decisions for the same protocol instance. Inductive methods were first applied to security protocol analysis in [43] and used more recently in [17], both in the analysis of properties of authentication protocols [47]. Fairness for the GJM protocol was also proved independently by Das and Dill using predicate abstraction and the finite-state model checker, Murφ in [19].

One of the important obstacles was to define a signer's power to control the outcome of a protocol. Effectiveness and fairness are both trace-based properties and have been studied extensively in the literature [35, 45, 3]. In [45], the authors also study a signer's ability to control the outcome of a protocol. Ability to control the outcome of a protocol is not a trace-based property and is naturally stated in terms of game strategies. A protocol is said to be *balanced* for a signer if it does not provide the other signer the power to control the outcome.

We formally state and prove *balance for honest signers*: assuming that the intruder takes only finitely many steps and that a signer is honestly following the protocol, we prove that, at any stage of the protocol, the other signer, even a strongly dishonest one, does not have both the power to complete the exchange as well as the power to abort it.

More precisely, we view A's power to abort as A's ability to prevent the successful completion of the exchange in coalition with the intruder, regardless of the actions of the other (honest) signer and the actions of the trusted third party. Similarly, we view A's power to complete the exchange as A's ability to prevent the abort in coalition with the intruder, regardless of the actions of the other (honest) signer and the actions of the trusted third party. Because a signer cannot control the actions of other participants, we take the view that the only ability that a signer has is that it may choose to fire a certain subset of its allowable actions at any stage of the protocol and hope to determine the outcome in that way.

Let us emphasize that balance is not a trace-based property in that it refers to the entire execution tree rather than to any single branch. Balance may be stated equivalently in terms of certain recursive properties of finite trees, which we prove by inductive methods. This recursive characterization lead to automated verification of balance for signature-exchange protocols. As far as we know, this was the first attempt to formalize a signer's ability to control the outcome of the protocol [15]. Later on, balance was studied with respect to security of communication channels in [36].

The proof of balance for honest signers closely mimics the proof of effectiveness. One of the ingredients in the proof of effectiveness is the special properties of the communication channels to the trusted third party. Another key ingredient in the proof is that an honest signer may non-deterministically exercise any of the options that are available to it, including the ability to contact the trusted third party at any stage of the protocol.

As we argued before, this does not capture all the nuisances of the natural behavior of signers. For example, in order to avoid the extra cost, a signer may prefer to wait for its counterparty than rushing to the third party for error recovery. In order to capture the natural behavior of the signers, we refine our approach and give a model in which we can discuss general two party signature-exchange protocols and signer bias. This forms the second part of our thesis (chapters 4, 5 and 6).

The challenge of modeling signer bias is met in two stages. First, we equip our model with signals, which we call *timers*, that the signers use to decide when to contact the trusted third party. In our formalism timers should be thought of as local signals. They do not refer to any global synchronous time. Secondly, as a direct way of mathematically characterizing biased behavior, we allow a biased signer to give its opponent the chance to signal whether to wait for a message or contact the third party. As we shall explain shortly, this gives us a relatively easy way to define the set of traces associated with a signer's strategy, while staying within the traditional nondeterministic, untimed setting. Before, we explain this further, we discuss some aspects of our model and definitions of desired protocol properties.

Another challenge was to find a uniform way to discuss the various cryptographic primitives used in different signature-exchange protocols. For each cryptographic operation used in messages of a protocol, we assume there is some MSR characterization of the computability properties of this operation. This allows us to discuss the cryptographic primitives in a uniform way. However, we emphasize, that the results in this thesis apply to signature-exchange protocols only to the extent that this MSR characterization accurately reflects the properties of the primitives of the protocol. We have checked that MSR can accurately represent the black-box versions of the primitives in [3, 9, 28]. We discuss the characterization of the primitives of the Garay, Jakobsson and Mackenzie protocol [28] in chapter 2.

One of the challenges is to model dishonest behavior in the revised model. In our analysis of the GJM protocol, a dishonest signer shared its private key with the intruder. In the revised model, we depart from this. Instead, we model dishonest behavior by giving each of the signers some additional rules that reflect similar capabilities. These additional rules are collectively called the *threat model*.

In the threat model, we shall assume that the trusted third party is honest and always follows the protocol. There are some protocols, *e.g.* [27], which ensure that a limited misbehavior of the third party does not compromise certain security guarantees for honest signers. However, we shall not consider third party misbehavior in our analysis.

No assumptions are made on the communication channels to the trusted third party. These channels are protected according to the assumptions of the protocol. This protection is modeled in the threat model. We also limit our attention to single protocol runs. Since our impossibility result holds even for a single run, it shall also hold for concurrent runs.

In this revised model, we give definitions of *strategies* and coalitions amongst protocol participants. Using the definition of strategies, we give the definition of fairness, and show that this definition may be equivalently stated as reachability. Effectiveness is stated in terms of reachability.

One of the important definitions is that of an optimistic protocol. Recall that a protocol is said to be optimistic if the trusted third party is contacted only in the case of a dispute. In the literature, a protocol is said to be optimistic [28, 1, 2, 3], if there is a trace such that the signers can exchange signatures without contacting the trusted third party. However, this definition does not capture an inherent assumption: the signers should be patient for the other signer for this optimistic exchange to take place. We say that a protocol is *optimistic* for a signer, Alice, if it is the case that whenever Alice is willing to wait "long enough", then the other signer, Bob, has a strategy to exchange signatures without any of them contacting the third party. A protocol is said to be optimistic if it is optimistic for both signers. Alice's willingness to wait long enough is modeled by giving Bob the ability to signal Alice the option to contact the third party.

In this model, we define a signer's power to determine the outcome of the protocol. Assuming that a signer Alice is honest, we say that the other signer, Bob, has the *power to abort the exchange* if Bob has a strategy to prevent Alice from getting Bob's signature on the pre-agreed text. Bob is said to have *power to obtain Alice's signature* if Bob has a strategy to get Alice's signature. The definitions of power to abort the exchange and the power to obtain the adversary's signature can be extended to biased signers. We consider two kinds of biased signers: *optimistic* and *interested*.

Recall that an optimistic signer is one who waits for some period of time before contacting the trusted third party, and an interested signer is one who waits for some period of time before quitting the exchange or contacting the trusted third party with a request to abort the exchange. The definitions of power to abort and the power to obtain Bob's signature are easily extended against an optimistic Alice if we allow Bob to signal Alice the option to go ahead and contact the trusted third party. Against an interested Alice, Bob is allowed to signal Alice the option to quit or contact the third party for an abort. Apart from these signals, the only ability that Bob has is that Bob may choose to fire a certain subset of its allowable actions at any stage of the protocol and hope to determine the outcome in that way.

A signer Bob is said to have an *advantage* if at some stage in the protocol Bob has both the power to abort and the power to obtain Alice's signature. One of the main results that we show is that advantage cannot be eliminated in a two-party, fair and optimistic protocols. This result is similar in spirit to the well-known result on impossibility of distributed consensus in presence of faults [26] and our proof is a three-valued version of the proof in [26, 24]. Recall that earlier in the introduction, we had noted that a third party is need to achieve fairness for two-party signature exchange protocols [24, 42]. Our result shows that even if we add a third party in order to achieve fairness, the asymmetry of communication reappears in the form of an advantage: no two-party signature exchange protocol is balanced.

More precisely, if S is a state that describes the internal state of the signers, trusted third party, timers and the communication channels, then we affix a pair of values, win, one each for Alice and Bob. We say that Alice wins in S with value 2 if Alice has Bob's signature in S, she wins with value 1 if Alice does not have Bob's signature but may obtain it with the help of trusted third party, and Alice wins with value 0 if Alice cannot get Bob's signature, even with the help of the trusted third party. We show that in any fair and optimistic protocol, there is a state such that the win value of exactly one of the signers is 1, while that of the other signer is 0. In this state, signer with win value 0 looses its power to abort the exchange. The signer with win value 1 still has the power to abort the exchange and enjoys an advantage over its counterparty.

Since advantage cannot be eliminated, the best a protocol can do to protect signers is to prevent the party with the advantage from being able $\underline{\text{to prove}}$ to an outside observer that it has the advantage. Recall that a protocol is said to be abuse-free if the protocol provides this protection to both the signers. In related work with John Mitchell, Andre Scedrov and Vitaly Shmatikov [16], we formalize abuse-freeness using epistemic logic [25, 33]. However, this work lies outside the scope of this thesis and is not included here.

Related work Formal methods used for verification of fair exchange include FDR [32] and CSP [44]. Shmatikov and Mitchell used the finite-state model checker Murφ

in [45] to perform an extensive study of signature exchange protocols in [2, 28]. Das and Dill used Murφ and predicate abstraction to prove that the protocol in [28] is fair.

None of these techniques, however, can be used to express the adversarial nature of fair exchange and capture the nuisances of the bias of signers. Game theory has been previously applied to the study of signature-exchange protocols by Chadha *et al.* [15], and Kremer and Raskin [35, 36]. In these approaches, the focus is on formalizing fairness and balance for the strongest possible honest signer, without taking into account optimism and other natural biases of signers. Additionally in [36], an analysis of balance with respect to security of communication channels.

Another game-theoretic model of fair exchange was developed in [12], but it focuses mainly on economic equilibria in fair exchange. Even the cryptographic proofs of correctness by the protocol designers [1, 3, 28] focus on basic fairness and ignore the issues of optimism and fundamental asymmetry of communication between the signers and the trusted third party.

The thesis is organized as follows. In chapter 2, we present and define the revised two-party signature exchange protocol of Garay, Jakobsson and Mackenzie [28] using the multiset-rewriting formalism [14, 13, 22]. In chapter 3, we state and prove fairness, effectiveness, and balance for honest signers for the revised protocol. Various aspects of the work in chapters 2 and 3 were carried out with Andre Scedrov and Max Kanovich. The chapters 2 and 3 are an extended version of our paper that was presented at the Eighth ACM conference on Computer and Communications Security [15].

In chapter 4, we present our refined semantic framework and define a class of two-party signature-exchange protocols with trusted third party. In chapter 5, we give the formal definitions of fairness, effectiveness, optimism and advantage of signature exchange protocols in the refined model. In chapter 6, we study the relationships amongst different protocol properties and establish our fundamental impossibility result. Various aspects the work in chapters 4, 5 and 6 were carried out with John Mitchell, Andre Scedrov and Vitaly Shmatikov. Some of the results and definitions in chapters 4 and 5 will be announced in the Fourteenth International Conference on Concurrency Theory [16].

In chapter 7, we summarize our results and discuss future directions in research on signature-exchange protocols.

Chapter 2

The Garay-Jakobsson-MacKenzie Protocol (GJM)

In this chapter, we shall describe a protocol obtained by a slight revision of the two-party optimistic signature exchange protocol of Garay, Jakobsson, and MacKenzie (GJM) in [28, 40]. We shall discuss the differences in our protocol and these protocols in section 2.3. After discussing the protocol informally, we shall give background on multiset-rewriting formalism (MSR) in section 2.4. This formalism is used to formalize the revised protocol in section 2.5.

Before, we start describing the protocol, we shall discuss some properties of the cryptographic primitives used in the GJM protocol. These properties are shared by several signature-exchange protocols [3, 28, 9] and are important in ensuring abuse-freeness. Later on, we shall discuss the cryptographic primitives of the GJM protocol in detail.

Signature Commitments. The fundamental notion underlying optimistic signature exchange protocols is that of a *verifiable and convertible signature commitment*. Informally, a signature commitment is used by each signer to convince its counterparty 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 signature commitment may allow the creator of the commitment to designate a verifier [34]: nobody other than the designated verifier will be convinced of the identity of the creator.

Signature commitments are designed in a way that allows the creator of the commitment to designate a converter who may convert the commitment into a universally verifiable signature [8]. Typically, a third party trusted by both signers 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. It then uses commitments exchanged by the signers to resolve the protocol fairly.

Let $vcsc_A(m, B, T)$ abstractly denote the verifiable, convertible signature commitments used in the protocols in [3, 9, 28]. The common properties showed by $vcsc_A(m, B, T)$ are:

a) $vcsc_A(m, B, T)$ is created 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 [3, 9, 28], 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 $sig_A(m)$ by T.

An outside observer C cannot distinguish a genuine $vcsc_A(m, B, T)$ from B's simulation. Hence, C does not know simply from $vcsc$ that A is participating in the protocol and $vcsc$ cannot be used by B as a proof of A's participation. This property is crucial in ensuring abuse-freeness.

We are now ready to describe the revised protocol. We start by describing the purpose, cryptographic assumptions, communication model under which the protocol is meant to be executed.

2.1 Purpose and assumptions of the protocol

The purpose of the protocol is to enable two parties, O and R, to exchange signatures on a previously agreed upon text, m with the help of a trusted third party, T. The protocol is designed to ensure effectiveness, fairness and abuse-freeness.

Each protocol participant is assumed to have a private signing key and a corresponding public verification key. Each participant is identified with this private signing/public verification key pair. In particular, for the rest of this section if we say that "*A can..*", we mean anyone that possesses the private signing key of A.

The protocol uses a cryptographic primitive, *private contract signature*(PCS) introduced in [28]. We write $PCS_O(m, R, T)$, for the private contract signature of O on a preagreed text m, intended for R with respect to T. PCS is the realization of the verifiable, convertible, signature commitment, *vcsc*. The main properties of private contract signatures are:

1. $PCS_O(m, R, T)$ can be computed by O.

2. There is a probabilistic polynomial−time algorithm $PCS{-}Ver$ such that $PCS{-}Ver(m, O, R, T, S) = \mathit{true}$ if S= $PCS_O(m, R, T)$. This algorithm can be run by anybody who knows the public verification keys of O, R and T.

3. R can compute $S = FakeSign_R(m, O, T)$ such that $PCS{-}Ver(m, O, R, T, S)$ $= \mathit{true}$. Only O or R can compute S such that $PCS{-}Ver(m, O, R, T, S) =$ true. O can verify whether it was created by R and similarly R can verify whether it was created by O. T can also verify whether it was generated by O or R. This is R's simulation of $PCS_O(m, R, T)$.

4. O can convert $PCS_O(m, R, T)$ into a conventional digital signature, $sig_O(m)$. There is a probabilistic-polynomial time algorithm, $S{-}Ver$, such that $S{-}Ver(m, O, T, S) = \mathit{true}$ iff $S = sig_O(m)$. Nobody other than O can compute S such that $S{-}Ver(m, O, T, S) = \mathit{true}$.

5. T can convert $PCS_O(m, R, T)$ into a digital signature, $TP{-}Sig_O(m)$. There is a probabilistic-polynomial time algorithm, $T{-}Ver$ such that $T{-}Ver(m, O, T,$ $S) = \mathit{true}$ iff $S = TP{-}Sig_O(m)$. Nobody other than T can compute S such that $T{-}Ver(m, O, T, S) = \mathit{true}$.

Our formalization of the cryptographic primitives will involve an abstract version of these primitives.

We shall assume an asynchronous communication model. An *intruder* may assume full control over the communication channel between the two signing parties, O and R. The intruder may intercept messages on these channels, decompose them, compose new messages and send these messages on the channels.

However, the signers have separate communication channels to T. These channels are assumed to be *write-protected*, i.e., nobody except the T and the signer (or anybody who possesses the private signing key of the signer) can write on this channel. Furthermore, these channels are assumed to be *transparent*; the channel never loses a message, unless the intended party reads it; and the intruder can observe messages from this channel without blocking or delaying them. We have relaxed the condition of private channels as stated in [28].

In our analysis, a signer may be dishonest and deviate from the protocol. We shall discuss a *strongly dishonest* signer which shares its private keys with the intruder prior to the execution of the protocol. On the other hand, we assume that the trusted third party is well-behaved and does not play the role of a party interested in exchanging signatures.

T is assumed to maintain a permanent database of each of the protocol instances that it has acted upon before. In our analysis, we further assume that T does not misbehave and does not act as a party to a signature-exchange. In [40], the authors allowed T to be accidently corrupt, namely the database of T was visible to the intruder. Since we have relaxed the notion of private channels, we do allow the accidently corrupt behavior of T, by making all communication between it and the signers observable to the intruder.

2.2 Protocol description

The protocol itself consists of three subprotocols: *exchange*, *abort*, and *resolve* subprotocols. Usually the parties would try to achieve the exchange by executing the exchange subprotocol. They would contact T using one of the other two subprotocols when they think something is amiss. Once they contact it, they no longer take part in the exchange subprotocol.

In the protocol description, when a signer A sends a message, *mssg* intended for B, it will be abbreviated as $A \rightarrow B : mssg$.

In the description, the two parties will be designated as O, the originator and R, the respondent. Before executing the protocol, the signers are assumed to have agreed upon each other's identity, the text m and the identity of the trusted third party, T. As in [28, 40], we do not consider how this is achieved in our analysis. In addition, they also agree on a globally unique identifier n, before they execute the protocol. Different instances of the protocol would have different identifiers. We further assume that the intruder knows this identifier. Let $pd =< m, n, O, R, T >$. We shall also assume that the globally unique identifier is registered with T. Please note that T does not need to be involved in this registration: there might be just a secondary registration entity that maintains the list of registered pd. We begin by describing the exchange subprotocol.

Exchange subprotocol: O displays its commitment to sign by sending $PCS_O(pd, R, T)$, intended for R. If R receives it, then it displays its commitment by sending $PCS_R(pd, O, T)$, intended for O. If O receives it, it sends $sig_O(pd)$, intended for R. If R receives it, it sends $sig_R(pd)$ intended for O and the protocol finishes for R. The protocol finishes for O when it receives $sig_R(pd)$. The protocol steps are described as

$O \rightarrow R : me_1{=}PCS_O(< m, n, O, R, T >, R, T)$

$R \rightarrow O : me_2{=}PCS_R(< m, n, O, R, T >, O, T)$

$O \rightarrow R : me_3{=}sig_O(< m, n, O, R, T >)$

$R \rightarrow O : me_4{=}sig_R(< m, n, O, R, T >)$

Abort Subprotocol: O may request T to abort the protocol after it sends $PCS_O(pd, R, T)$ and before it receives $PCS_R(pd, O, T)$. This it does by sending $ma_1 = sig_O(abort, pd)$ on the $O{-}T$ channel. T on receiving the abort request, looks in its database and checks if it has ever answered a request for pd that it received before on the $O{-}T$ Channel. If it has, it will not send back anything, otherwise it checks if it has resolved pd before. If it has, it sends the stored resolution $< TP{-}Sig_O(pd),\ TP{-}Sig_R(pd) >$. Otherwise it issues an abort_token, $sig_T(ma_1)$, and sends it on the $O{-}T$ channel. It raises its abort flag for pd and stores the abort_token. An abort_token is a promise by T that it has not and will not resolve pd in future.

While R is not allowed to abort, it is allowed to quit before it receives the first message.

Resolve Subprotocol: O may request T to resolve the protocol after it sends $sig_O(pd)$ and before it receives $sig_R(pd)$. R may run the subprotocol after it sends $PCS_R(pd, O, T)$ and before it receives $sig_O(pd)$.

R requests T to resolve the protocol instance by sending $(PCS_O(pd, R, T), PCS_R(pd, O, T))$. T on receiving the resolve request, checks if it has ever answered a request for pd that it received before on the $R{-}T$ channel. If it has, it will not send back anything, otherwise it checks if it has aborted or resolved pd before. If it has been aborted or resolved before then it sends the stored decision on the $R{-}T$ channel. If it has neither aborted nor resolved the protocol, it converts the PCS to $TP{-}Sig_O(pd)$, $TP{-}Sig_R(pd)$ and sends it on the $R{-}T$ channel. It raises its resolved flag for pd and stores the resolution.

O's resolve subprotocol is similar.

The authors of the original protocol [28] claimed several properties, including fairness and abuse-freeness. We discuss these properties in chapter 3.

2.3 Differences from the original protocol

In the original protocol [28], the resolve request consisted of the requesting party's signature on the text and the private contract signature of the other party. In [40], the authors showed that in case T is accidentally corrupt, this may lead to loss of fairness. They suggested a fix in which the resolve request consisted of private contract signatures of both the parties. We take this fixed protocol as our reference point and discuss our differences.

We discuss briefly the differences in our protocol from the versions of the GJM protocol as defined in [28, 40]:

1. We assume perfect cryptography, while in [28], the cryptographic primitives, are defined in terms of probabilistic polynomial time computations.

2. We assume write-protected and transparent channels between the signers and T, instead of private channels.

3. We explicitly include the identities of the parties involved in the exchange in every message that is being sent. This is in accordance with the well-established practice of including the participants' identities in each step of a cryptographic protocol. Also, now each exchange includes a globally unique identifier. In the previous two versions, there was no such identifier. We did this because both [28, 40] allow the following scenario:

   O and R try to sign a pre-agreed text m and trusted third party, T. O sends

$PCS_O(m, R, T)$ intended for R. The intruder, henceforth called I listens on the channel between O and R.

$O \rightarrow R$: me_1=$PCS_O(m, R, T)$

I intercepts it. R times out waiting for a reply, and quits. O asks T for an abort_token, who not having an entry for the exchange before, issues the abort_token.

$O \rightarrow T$: ma_1=$sig_O(abort, m, O, R, T)$

$T \rightarrow O$: ma_2=$sig_T(sig_O(abort, m, O, R, T))$

Now, suppose O and R decide to sign m again with T as the trusted third party. They start by using the exchange protocol.

$O \rightarrow R$: me_1=$PCS_O(m, R, T)$

$R \rightarrow O$: me_2=$PCS_R(m, O, T)$

$O \rightarrow R$: me_3=$sig_O(m)$

$R \rightarrow O$: me_4=$sig_R(m)$

I intercepts.

O does not receive $sig_R(m)$ and times out. Then O asks T for a resolution, which having an abort_token in its database for this (m, O, R, T) sends the abort_token to O. Hence R has O's signature on *pd* and O only an abort_token, thus violating fairness.

4. The protocol as presented in [28] seems to allow T to be contacted multiple times by each party. This seems to violate an optimistic setting since a mali-

cious signer could contact T multiple times forcing it to do expensive database searches. We revise the protocol so that T acts on only one request from each signer for each run of the protocol. This should be sufficient because according to the protocol definition each signer is allowed to contact T only once. O may contact it for an abort or a resolve. R may contact it for a resolve.

2.4 Background on multiset-rewriting formalism (MSR)

The protocol formalism we use to define the protocol actions is multiset-rewriting with existential quantification, MSR, as described in [14, 13, 22]. This formalism can be seen either as an extension of some standard models of computation, *e.g.*, multiset transformation [4] and chemical abstract machine [6], or as the Horn fragment of Linear Logic [31].

We outline the formalism here briefly. Its syntax involves terms, facts and rules. If one wants to represent a system in this notation, one begins by choosing a *first-order vocabulary.* This is a standard notion from many-sorted algebra or first-order logic [23]. 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. This means that a fact is the result of applying a predicate symbol to ground terms of the correct sorts.

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

As an example, consider the state, $\{P(f(a)), P(b)\}$. Consider the rule $P(x) \longrightarrow \exists z.Q(f(x), z)$. A possible next state is obtained by instantiating the above rule to $P(f(a)) \longrightarrow \exists z.Q(f(f(a)), z)$. Applying this rule, we choose a new value, c, for z and replace $P(f(a))$ by $Q(f(f(a)), c)$. This gives us the state $\{Q(f(f(a)), c), P(b)\}$.

As presented in [14, 13, 22], a protocol theory consists of three parts: a bounded phase describing protocol initialization that distributes keys or establishes other shared information, a role generation theory that designates possibly multiple roles that each principal may play in a protocol (such as initiator, responder, client, or server), and a disjoint union of bounded subtheories that each characterize a possible role, and which ensure that each role is finite. Furthermore, the multiset-rewriting formalism allows us to formulate one standard intruder theory per given signature, which describes any intruder for any protocol formalized in the language of the given signature. Looping is prevented by certain technical conditions discussed in [14, 13, 22].

In our discussion we shall need the following definition [14, 13, 22]:

Definition 2.4.1. A rule $l \to r$ *consumes* P *facts* if some atomic formula $P(\vec{t})$ occurs more times in l than in r. A rule $l \to r$ *creates* P *facts* if some atomic formula $P(\vec{t})$ occurs more times in r than in l.

In our protocol, the participants would be identified with their private signature keys and the corresponding public verification keys. Usually a digital signature on a message is a pair consisting of a message and the signature on the message. The signature is used to verify the person who is supposed to have generated it. In our formalism a message x, signed with a private key k_s is denoted by $sig(k_v, x)$ where k_v is the corresponding public verification key. This notation, allows us to model the verification by pattern matching. For example, a protocol participant, A, who is waiting for a message signed under a key whose public verification key is k_v when presented with a message, $sig(k_v, x)$ on the network would accept it. This rule expressed as $A(k_v), N(sig(k_v, x)) \to A(k_v, sig(k_v, x))$ would abstract away the verification process. This verification by pattern matching is similar to the encryption by pattern matching as presented in [14, 13, 22].

Given a participant O with public verification key k_o, a participant R with public verification key k_r and a trusted third party, T with public verification key k_t, $PCS_O(x, R, T)$ will be denoted by $PCS(k_o, x, k_r, k_t)$. $FakeSign_R(m, O, x)$ by $FakeSign(k_r, x, k_o, k_t)$ and $TP{-}Sig_O(x)$ by $tsig(k_t, k_o, x)$.

2.5 Protocol Definition in MSR

We now discuss the precise definition of the protocol in MSR. We start by discussing some of the sorts used in the definition.

Sorts. We shall assume that our vocabulary includes a sort *mssg* that is used for messages that can be sent by the protocol participants and the intruder. The vocabulary shall also contain two further sorts: *private_key* and *public_key*. The sort *private_key* is used for private signing keys for the principals and the sort *public_key* is used for public verification keys for the principals. We use $\mathsf{k}, \mathsf{k}', \mathsf{k}_\mathsf{a}, \ldots$ to range over variables of the sorts *private_key* and *public_key*, and $k, k', k_a, k_1, \ldots$ to range over values of the sorts *private_key* and *public_key*. We use $\mathsf{m}_1, \mathsf{m}_2, \ldots$ to range over variables of the sort *mssg* and $m_1, m_2, \ldots$ to range over values of the sort *mssg*. There are three special constants of the sort *mssg* that are used in the protocol messages: *abort*, *aborted* and *resolved*.

We assume that our vocabulary also contains further two sorts: *preagreed_text* and *unique_identifier* as sub-sorts of the sort *mssg*. The sort *preagreed_text* is used for the pre-agreed texts. We use $\mathsf{m}, \mathsf{m}', \ldots$ is used to range over variables of the sort *preagreed_text* and $m, m', \ldots$ are used to range over values of the sort *pre−agreed_text*. The sort *unique_identifier* is used for globally unique identifiers. We use n, n' to range over variables of the sort *unique_identifier* and $n, n', \ldots$ to range over values of the sort *unique_identifier*.

We shall assume that our vocabulary consists of the following function symbols: *sig*, *PCS*, *tsig* and *FakeSign*. The sorts of these function symbols are:

$$\begin{aligned}
sig:&\quad public\_key \times mssg \rightarrow mssg\\
PCS:&\quad public\_key \times mssg \times public\_key \times public\_key \rightarrow mssg\\
tsig:&\quad public\_key \times public\_key \times mssg \rightarrow mssg\\
FakeSign:&\quad public\_key \times mssg \times public\_key \times public\_key \rightarrow mssg
\end{aligned}$$

Our vocabulary also consists of a function symbol for pairing messages:

$$< \_, \_ >: mssg \times mssg \rightarrow mssg$$

In our description, we shall use $< x_1, x_2, x_3 >$ as a short hand for $<< x_1, x_2 >, x_3 >$, $< x_1, x_2, x_3, x_4 >$ as a short hand for $<<< x_1, x_2 >, x_3 >, x_4 >$, and so on.

We assume that our vocabulary consists of three unary predicate symbols D, M, and C whose argument is of the sort *mssg*. D is used for decomposable messages known to intruder, M is used for information stored in intruder "memory", and C is composable messages known to intruder.

We shall discuss other predicate symbols as we encounter them in our definition. Now, we are ready to give the MSR definition of the protocol. Let $\mathsf{m}, \mathsf{n}, \mathsf{k_o}, \mathsf{k_r}, \mathsf{k_t}$ be variables of the sort *preagreed_text*, *unique_identifier*, *public_key*, *public_key* and *public_key* respectively. For the sake of convenience, we use some abbreviations: $\mathsf{pd} =< \mathsf{m}, \mathsf{n}, \mathsf{k_o}, \mathsf{k_r}, \mathsf{k_t} >$, $\mathsf{me_1} = PCS(\mathsf{k_o}, \mathsf{pd}, \mathsf{k_r}, \mathsf{k_t})$, $\mathsf{me_2} = PCS(\mathsf{k_r}, \mathsf{pd}, \mathsf{k_o}, \mathsf{k_t})$,$\mathsf{me_3} = sig(\mathsf{k_o}, \mathsf{pd})$, $\mathsf{me_4} = sig(\mathsf{k_r}, \mathsf{pd})$, $\mathsf{ma_1} = sig(\mathsf{k_o}, < \mathsf{abort}, \mathsf{pd} >)$, $\mathsf{mr_1} =< \mathsf{me_1}, \mathsf{me_2} >$, $\mathsf{ab\_tok} = sig(\mathsf{k_t}, \mathsf{ma_1})$, and $\mathsf{res\_cn} =< tsig(\mathsf{k_t}, \mathsf{k_o}, \mathsf{pd}), tsig(\mathsf{k_t}, \mathsf{k_r}, \mathsf{pd}) >$.

Intuitively, pd identifies a protocol instance with the globally unique identifier n, the pre-agreed text m, and the roles of O, R and T being played by principals whose public keys are k_o, k_r and k_t respectively. $\mathsf{me_i}$ is the $i-th$ message of the exchange protocol for the protocol instance *pd*. $\mathsf{ma_1}$ is the abort request for *pd*, ab_tok is the abort_token for *pd*, $\mathsf{mr_1}$ is the resolve request for *pd* and res_cn is the resolution for *pd* from T.

Initial set of facts. We assume an initial finite set (not a multiset) of facts, Σ. We assume that we have the following binary predicate symbols in our vocabulary: KP, *BadKey*, *HonestGuy* and TTP. The first argument of these predicates is of the sort *private_key* and the second argument is of the sort *public_key*. We also assume that we have unary predicate symbols *AnnK*, *AnnT* and *Preagreed*. The argument of *AnnK* and *AnnT* is of the sort *public_key*, and of *Preagreed* is of the sort *preagreed_text*.

Σ contains many $KP(k_s, k_v)$ predicates. A $KP(k_s, k_v)$ predicate identifies a signer, whose private/public key pair is (k_s, k_v). Furthermore, for any two facts $KP(k_1, k_2)$,
$KP(k_3, k_4) \in \Sigma$, k_1 is not the same as k_3 and k_2 is not the same as k_4.

Honest signers amongst these are identified by $HonestGuy(k_s, k_v)$ predicates. Strongly dishonest signers are identified by $BadKey(k_s, k_v)$ predicates. Both keys in the $BadKey$ predicate are known to the intruder, *i.e.*, if $BadKey(k_s, k_v) \in \Sigma$ then $M(k_s), M(k_v) \in \Sigma$. For each $KP(k_s, k_v) \in \Sigma$, we have $\underline{\text{either}}$ $HonestGuy(k_s, k_v) \in \Sigma$ $\underline{\text{or}}$ $BadKey(k_s, k_v) \in \Sigma$. There are no other $HonestGuy$ and $BadKey$ facts in Σ.

Σ also contains $TTP(k_{ts}, k_t)$ predicates which identifies trusted third parties with private/public signing key pairs (k_{ts}, k_t). Furthermore, for any two facts $TTP(k_1, k_2)$, $TTP(k_3, k_4) \in \Sigma$, k_1 is not the same as k_3 and k_2 is not the same as k_4. Furthermore, the key pairs amongst the KP and TTP predicates are pair-wise disjoint, *i.e.*, if $KP(k_1, k_2) \in \Sigma$ and $TTP(k_3, k_4) \in \Sigma$ then k_1 is not the same as k_3 and k_2 is not the same as k_4.

The publicly announced verification keys of the signers are denoted by $AnnK(k_v)$ predicates and the publicly announced verification keys of trusted third parties are denoted by $AnnT(k_t)$ predicates. For each $KP(k_s, k_v) \in \Sigma$, we must have $AnnK(k_v) \in \Sigma$, and for each $TTP(k_{ts}, k_t) \in \Sigma$, we must have $AnnT(k_t) \in \Sigma$. There are no other $AnnK$ and $AnnT$ facts.

Σ also contains $Preagreed(m)$ predicates which identifies pre-agreed texts. We further assume that the intruder knows all these texts, *i.e.*, if $Preagreed(m) \in \Sigma$ then $M(m) \in \Sigma$. There are no other facts in the set Σ.

The MSR definition also consists of a role generation rule, a theory each for O,R, T and the intruder.

2.5.1 Role Generation Theory

We assume that our vocabulary consists of three unary predicates O_0, R_0 and T_0 whose argument is of the sort *mssg*. These predicates are initial states for the protocol theories of O, R and T respectively. The role generation theory consists of a single rule:

$$KP(\mathsf{k_{os}}, \mathsf{k_o}), KP(\mathsf{k_{rs}}, \mathsf{k_r}), TTP(\mathsf{k_{ts}}, \mathsf{k_t}), \mathit{Preagreed}(\mathsf{m}) \rightarrow \exists n.O_0(\mathsf{pd}), R_0(\mathsf{pd}), T_0(\mathsf{pd}),$$
$$KP(\mathsf{k_{os}}, \mathsf{k_o}),\ KP(\mathsf{k_{rs}}, \mathsf{k_r}),\ TTP(\mathsf{k_{ts}}, \mathsf{k_t}), \mathit{Preagreed}(\mathsf{m}), M(\mathsf{n})$$

This rule shall be henceforth referred to as RG. In the role generation theory, two principals with key pairs $(\mathsf{k_{os}}, \mathsf{k_s})$ and $(\mathsf{k_{or}}, \mathsf{k_r})$ agree upon text m, identity of the trusted third party $\mathsf{k_t}$, and a globally unique identifier, n. The uniqueness of n is guaranteed because existential quantification means generation of a fresh value. The trusted third party's state is initialized to $T_0(\mathsf{pd})$. The state $T_0(\mathsf{pd})$ should be thought of as the registration of pd. It should be thought of the state of T in which T has no entry for pd in its database. This intuition makes sense because of the freshness of n, T could not have heard of pd before this rule is applied. This rule abstracts away the agreement of text, identity of T and the globally unique identifier. The predicate M indicates the intruder's memory. Now the signers are ready to commence the protocol.

2.5.2 Protocol Theory

The protocol theory for O is shown in Table 2.1.

O: Protocol Theory for O

O_1 : $O_0(\mathsf{pd}) \rightarrow O_1(\mathsf{pd}, \mathsf{me}_1), N(\mathsf{me}_1)$

$O_{ab?}$: $O_1(\mathsf{pd}, \mathsf{me}_1) \rightarrow O_{ab?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{ma}_1), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{ma}_1)$

O_2 : $O_1(\mathsf{pd}, \mathsf{me}_1), N(\mathsf{me}_2) \rightarrow O_2(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2)$

O_3 : $O_2(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2) \rightarrow O_3(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3), N(\mathsf{me}_3)$

$O_{res?}$: $O_3(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3) \rightarrow$

$O_{res?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{mr}_1), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{mr}_1)$

O_{com} : $O_3(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3), N(\mathsf{me}_4) \rightarrow O_{com}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{me}_4)$

O_{ab1} : $O_{ab?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{ma}_1), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{ab\_tok}) \rightarrow O_{ab1}(\mathsf{pd}, \mathsf{me}_1, \mathsf{ma}_1, \mathsf{ab\_tok})$

O_{res1} : $O_{ab?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{ma}_1), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{res\_cn}) \rightarrow O_{res1}(\mathsf{pd}, \mathsf{me}_1, \mathsf{ma}_1, \mathsf{res\_cn})$

O_{ab2} : $O_{res?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{mr}_1), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{ab\_tok}) \rightarrow$

$O_{ab2}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{mr}_1, \mathsf{ab\_tok})$

O_{res2} : $O_{res?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{mr}_1), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{res\_cn}) \rightarrow$

$O_{res2}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{mr}_1, \mathsf{res\_cn})$

Table 2.1: Protocol theory for O

In the protocol definition, O has 10 states modeled by predicate symbols: O_i for $i \in \{0, 1, 2, 3, com, ab?, res?, ab1, res1, ab2, res2\}$. The sorts of these predicate symbols are clear from the context. The numbered states O_0, O_1, O_2, O_3 denote O's state during the exchange subprotocol execution. O_0 is the state of O at the start of the protocol. O_1 corresponds to the state in which it has sent me_1, O_2 corresponds to the state in which it has received me_2, O_3 to the state in which it has sent me_3. O_{com} corresponds to a state in which it has received me_4. $O_{ab?}$ corresponds to the state in which O has issued an abort request to T and O_{ab1} corresponds to the state in which it has an abort_token for this request and O_{res1} corresponds to the state in which it receives a resolution for this request. $O_{res?}$ corresponds to the state in which O has issued a resolve request to T and O_{ab2} and O_{res2} correspond to the states in which it has received an abort_token or a resolution, respectively for this request. As explained in section 2.4, these rules abstract the verification process of the cryptographic primitives.

The network between O and R is modeled by N predicate symbols. N is a unary predicate symbol whose argument is of the sort $mssg$. The write-protected transparent channel between O and T is modeled by ternary Rn predicates whose first argument is $\mathsf{k_o}$ and the second argument is $\mathsf{k_t}$. From here on, the protocol theory for O shall be denoted by $\mathbf{O}$.

As mentioned before, a strongly dishonest O is modeled by a $BadKey(\mathsf{k_s}, \mathsf{k_v})$ predicate. Both k_s, k_v are known to the intruder.

The protocol theory of T is given in Table 2.2, for R in Table 2.3. The protocol theory of T will be denoted by $\mathbf{T}$, R will be denoted by $\mathbf{R}$.

The intruder theory is given in divided into two tables 2.4 and 2.5. The intruder theory will be denoted by $\mathbf{I}$.

The intruder rules can be divided into three main groups: input/output, decomposition and composition rules. The rule REC allows the intruder to intercept messages on the network, rule REC_R allows the intruder to read messages on the channels to trusted third party. The rule SND allows the intruder to send messages on the network, SND_R and SND_T allow the intruder to send messages on the special channels to the trusted third party if it knows the private signing key of either the signer or the trusted third party.

The decomposition rule $DCMP$ allows the intruder to split pairs. The rule $READPCS$ allows the intruder to extract the message in PCS. The rules $ReadSig$, $ReadFakeSig$ and $ReadTPSig$ allow the intruder to extract the message in sig, $FakeSign$ and $tsig$ respectively. The rule LRN allows the intruder to copy messages from the decomposition memory to its memory.

The composition rule USE allows the intruder to copy messages from the memory to composition rules. The composition rule CMP allows the intruder to allow the intruder to make pairs. The rules PCS, Sig, $FakeSig$ and $TPSig$ allow the intruder to make PCS, sig, $FakeSign$ and $tsig$ if it knows the private keys. These rules are the formalization of the black-box version of the cryptographic primitives

T: Protocol Theory for T

T_{ab} : $Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{ma_1}), T_0(\mathsf{pd}) \rightarrow T_{ab}(\mathsf{pd}, \mathsf{aborted}, \mathsf{ab\_tok}), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{ab\_tok})$

T_{or} : $Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{mr_1}), T_0(\mathsf{pd}) \rightarrow T_{or}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{res\_cn})$

T_{rr} : $Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{mr_1}), T_0(\mathsf{pd}) \rightarrow T_{rr}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}), Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{res\_cn})$

T_{abf}: $Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{ma_1}), T_{ab}(\mathsf{pd}, \mathsf{aborted}, \mathsf{ab\_tok}) \rightarrow$

$T_{abf}(\mathsf{pd}, \mathsf{aborted}, \mathsf{ab\_tok}), Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{ab\_tok})$

T_{orf} : $Rn(\mathsf{k_t}, \mathsf{k_t}, \mathsf{mr_1}), T_{or}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}) \rightarrow$

$T_{orf}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}), Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{res\_cn})$

T_{rrf1} : $Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{ma_1}), T_{rr}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}) \rightarrow$

$T_{rrf1}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{res\_cn})$

T_{rrf2} : $Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{mr_1}), T_{rr}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}) \rightarrow$

$T_{rrf2}(\mathsf{pd}, \mathsf{resolved}, \mathsf{res\_cn}), Rn(\mathsf{k_o}, \mathsf{k_t}, \mathsf{res\_cn})$

Table 2.2: Protocol theory for T

of the protocol. The rule GEN allows the intruder to generate new arbitrary messages.

R: Protocol Theory for R

R_{quit} :	$R_0(\mathsf{pd}) \rightarrow R_{quit}(\mathsf{pd})$
R_1 :	$R_0(\mathsf{pd}), N(\mathsf{me}_1) \rightarrow R_1(\mathsf{pd}, \mathsf{me}_1)$
R_2 :	$R_1(\mathsf{pd}, \mathsf{me}_1) \rightarrow R_2(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2), N(\mathsf{me}_2)$
$R_{res?}$:	$R_2(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2) \rightarrow R_{res?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{mr}_1), Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{mr}_1)$
R_3 :	$R_2(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2), N(\mathsf{me}_3) \rightarrow R_3(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3)$
R_{com} :	$R_3(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3) \rightarrow R_{com}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{me}_4), N(\mathsf{me4})$
R_{ab} :	$R_{res?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{mr}_1), Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{ab\_tok}) \rightarrow$
	$R_{ab}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{mr}_1, \mathsf{ab\_tok})$
R_{res} :	$R_{res?}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{mr}_1), Rn(\mathsf{k_r}, \mathsf{k_t}, \mathsf{res\_cn}) \rightarrow$
	$R_{res}(\mathsf{pd}, \mathsf{me}_1, \mathsf{me}_2, \mathsf{me}_3, \mathsf{mr}_1, \mathsf{res\_cn})$

Table 2.3: Protocol theory for R and weakly dishonest R

I: Protocol Theory for the intruder

$I/ORules$:	
REC :	$N(\mathsf{x}) \to D(\mathsf{x})$
SND :	$C(\mathsf{x}) \to N(\mathsf{x})$
REC_R :	$Rn(\mathsf{k_1}, \mathsf{k_2}, \mathsf{x}) \to D(\mathsf{x}), Rn(\mathsf{k_1}, \mathsf{k_2}, \mathsf{x})$
SND_R :	$C(\mathsf{x}), M(\mathsf{k_{2s}}), KP(\mathsf{k_{2s}}, \mathsf{k_2}), ANNK(\mathsf{k_1}) \to$
	$Rn(\mathsf{k_2}, \mathsf{k_1}, \mathsf{x}), M(\mathsf{k_{2s}}), KP(\mathsf{k_{2s}}, \mathsf{k_2}), ANNK(\mathsf{k_1})$
SND_T :	$C(\mathsf{x}), M(\mathsf{k_{2s}}), TTP(\mathsf{k_{2s}}, \mathsf{k_2}), ANNT(\mathsf{k_1}) \to$
	$Rn(\mathsf{k_1}, \mathsf{k_2}, \mathsf{x}), M(\mathsf{k_{2s}}), TTP(\mathsf{k_{2s}}, \mathsf{k_2}), ANNT(\mathsf{k_1})$
$DecompositionRules$:	
$DCMP$:	$D(\langle \mathsf{x}, \mathsf{y} \rangle) \to D(\langle \mathsf{x}, \mathsf{y} \rangle), D(\mathsf{x}), D(\mathsf{y})$
LRN :	$D(\mathsf{x}) \to M(\mathsf{x})$
$ReadPCS$:	$D(PCS(\mathsf{k_o}, \mathsf{x}, \mathsf{k_r}, \mathsf{k_t})) \to D(PCS(\mathsf{k_o}, \mathsf{x}, \mathsf{k_r}, \mathsf{k_t})), D(\mathsf{x})$
$ReadSig$:	$D(sig(\mathsf{k_s}, \mathsf{x})) \to D(sig(\mathsf{k_s}, \mathsf{x})), D(\mathsf{x})$
$ReadFakeSig$:	$D(FakeSign(\mathsf{k_r}, \mathsf{x}, \mathsf{k_o}, \mathsf{k_t})) \to D(FakeSign(\mathsf{k_r}, \mathsf{x}, \mathsf{k_o}, \mathsf{k_t})), D(\mathsf{x})$
$ReadTPSig$:	$D(tsig(\mathsf{k_t}, \mathsf{k_o}, \mathsf{x})) \to D(tsig(\mathsf{k_t}, \mathsf{k_o}, \mathsf{x})), D(\mathsf{x})$

Table 2.4: Two-Phase Intruder Theory: Decomposition Rules

I: Protocol Theory for the intruder (continued)

$CompositionRules:$

$COMP:$	$C(\mathsf{x}), C(\mathsf{y}) \rightarrow C(\mathsf{x}), C(\mathsf{y}), C(\langle \mathsf{x}, \mathsf{y} \rangle)$
$USE:$	$M(\mathsf{x}) \rightarrow C(\mathsf{x}), M(\mathsf{x})$
$PCS:$	$M(\mathsf{k_{os}}), C(\mathsf{x}), KP(\mathsf{k_{os}}, \mathsf{k_o}), ANNK(\mathsf{k_r}), ANNT(\mathsf{k_t}) \rightarrow M(\mathsf{k_{os}}),$ $C(PCS(\mathsf{k_o}, \mathsf{x}, \mathsf{k_r}, \mathsf{k_t})), KP(\mathsf{k_{os}}, \mathsf{k_o}), ANNK(\mathsf{k_r}), ANNT(\mathsf{k_t})$
$FakeSign:$	$M(\mathsf{k_{rs}}), C(\mathsf{x}), KP(\mathsf{k_{rs}}, \mathsf{k_r}), ANNK(\mathsf{k_o}), ANNT(\mathsf{k_t}) \rightarrow C(\mathsf{x}),$ $M(\mathsf{k_{rs}}), C(FakeSign(\mathsf{k_r}, \mathsf{x}, \mathsf{k_o}, \mathsf{k_t})), KP(\mathsf{k_{rs}}, \mathsf{k_r}), ANNK(\mathsf{k_o}),$ $ANNT(\mathsf{k_t})$
$SIG:$	$M(\mathsf{k_{os}}), C(PCS(\mathsf{k_o}, \mathsf{x}, \mathsf{k_r}, \mathsf{k_t})), KP(\mathsf{k_{os}}, \mathsf{k_o}) \rightarrow M(\mathsf{k_{os}}),$ $C(PCS(\mathsf{k_o}, \mathsf{x}, \mathsf{k_r}, \mathsf{k_t})), C(sig(\mathsf{k_o}, \mathsf{x})), KP(\mathsf{k_{os}}, \mathsf{k_o})$
$TPSIG:$	$M(\mathsf{k_{ts}}), C(PCS(\mathsf{k_o}, \mathsf{x}, \mathsf{k_r}, \mathsf{k_t})), KP(\mathsf{k_{ts}}, \mathsf{k_t}) \rightarrow M(\mathsf{k_{ts}}),$ $C(PCS(\mathsf{k_o}, \mathsf{x}, \mathsf{k_r}, \mathsf{k_t})), C(tsig(\mathsf{k_t}, \mathsf{k_o}, \mathsf{x})), KP(\mathsf{k_{ts}}, \mathsf{k_t})$
$GEN:$	$\rightarrow \exists \mathsf{x}. M(\mathsf{x})$

Table 2.5: Two-Phase Intruder Theory: Composition Rules

Chapter 3

Properties of GJM protocol

In chapter 2, we gave a formal definition of a protocol that was obtained by the revision of the GJM protocol [28, 45]. The protocol definition consisted of an initial set of facts, Σ, a role generation rule RG, protocol theories **O**, **R** and **T** for each of the roles O, R and T respectively, and **I**, the theory for the intruder I. In this chapter we shall state and prove fairness and effectiveness for the revised protocol. In order to state and prove fairness we shall use a non-monotonic invariant on the states, *view of the intruder*. We shall then state and prove a few properties of the database of the trusted third party. These properties will be used in the proof of fairness. The proofs of fairness and effectiveness are carried out by inductive methods.

Assuming fairness, we shall state *balance for honest signers* for the revised protocol. Intuitively speaking, a protocol is balanced for a honest signer, if the other (possibly dishonest) signer cannot unilaterally decide the outcome of the protocol. We shall give a recursive characterization of this property and show that the protocol is balanced using this alternate characterization. The proof shall closely mimic the proof of effectiveness.

We shall need the following definition:

Definition 3.0.1. A *trace* from a state S_0 is a (possibly infinite) sequence of states $S_0, S_1, ...$ such that S_{i+1} is obtained from S_i by the application of the role generation rule RG or a transition rule in $\mathbf{O} \cup \mathbf{R} \cup \mathbf{T} \cup \mathbf{I}$. A state S' is said to be *reachable from* S if there is a finite trace from S leading to state S'. Any state reachable from the initial set of facts, Σ, is called a *reachable state.*

3.1 View of the intruder

In order to state the desirable properties and carry out their proofs we define a non-monotonic invariant, $W(S)$, on reachable states motivated by [17]. This invariant, which shall be called *the view of the intruder at* S, is the set of all possible messages that can be formed by the intruder using all the information available in network, the intruder memory, intruder's composition and decomposition state in the state S.

Definition 3.1.1. For a set of terms, Tms,

1. $analz(Tms)$ is the smallest superset, X, of Tms, such that

a. $u \in X$ and $v \in X$ if $< u, v > \in X$,

b. $u \in X$ if $sig(k, u) \in X$,

c. $u \in X$ if $PCS(k, u, k_1, k_t) \in X$,

d. $u \in X$ if $FakeSign(k, u, k_1, k_t) \in X$, and

e. $u \in X$ if $tsig(k, k_1, u) \in X$.

2. $synth(Tms)$ is the smallest superset, X, of Tms, such that

   a. $< u, v > \in X$ if $u \in x$ and $v \in X$,

   b. $PCS(k, u, k_1, k_t) \in X$ if $u, k^{-1} \in X$, where k^{-1} is the signature key corresponding to the verification key k, and

   c. $FakeSign(k, u, k_1, k_t) \in X$ if $u, k^{-1} \in X$, where k^{-1} is the signature key corresponding to the verification key k, and

   d. $sig(k, u) \in X$ if $x, k^{-1}, PCS(k, u, k_1, k_t) \in X$ where k^{-1} is the signature key corresponding to the verification key k.

   e. $tsig(k_t, k_1, u) \in X$ if $x, {k_t}^{-1}, PCS(k, u, k_1, k_t) \in X$ where ${k_t}^{-1}$ is the signature key corresponding to the verification key k.

3. For a given state S, the set of messages available to the intruder is defined to be $A(S) = \{u | C(u), D(u), M(u), N(u), Rn(k_1, k_2, u) \in S\}$. The *VIEW* of the intruder is defined to be $W(S) = synth(analz(A(S)))$. If $u \in W(S)$ then we say that u is in the view of intruder at S.

The intruder can possibly produce all these messages in future. Hence, as long as the intruder does not generate new data, its view does not change by any of its actions:

Proposition 3.1.2. *If S, S' are states such that S' is obtained from S by the application of rule in* $\mathbf{I}$ *other than GEN, then $W(S) = W(S')$.*

We state a few facts of this invariant and prove the above proposition in section 3.4. We shall also show in section 3.4 that an intruder does not learn the private signing keys of honest signers and trusted third parties. This will be needed to prove protocol properties:

Proposition 3.1.3. *Suppose Σ contains $HonestGuy(k, k^{-1})$ or $TTP(k, k^{-1})$. Then for all reachable states S, k is not in the view of the intruder at S.*

3.2 Effectiveness and fairness

We now state effectiveness and fairness in our formalism. Concurrent runs of the protocol are modeled by different instantiations of the role generation theory. We show that each instance in concurrent runs is fair and hence the protocol is fair. Assume that in state S_0, two principals O and R, whose private/public key pairs are (k_{os}, k_o), and (k_{rs}, k_r), agree to exchange signatures on the text m, and a trusted third party, T whose private/public key pair is (k_{ts}, k_t) by using rule RG. A fresh globally unique identifier, n gets generated and we have a new protocol instance identified by $pd = <m, n, k_o, k_r, k_t>$ Let the resulting state be S_1. For the sake of convenience, we use some abbreviations:

$pd = \langle m, n, k_o, k_r, k_t \rangle$, $me_1 = PCS(k_o, pd, k_r, k_t)$, $me_2 = PCS(k_r, pd, k_o, k_t)$, $me_3 = sig(k_o, pd)$, $me_4 = sig(k_r, pd)$, $ma_1 = sig_O(abort, pd)$, $mr_1 = \langle me_1, me_2 \rangle$, $ab\_tok = sig(k_t, ma_1)$, and $res\_cn = \langle tsig(k_t, k_o, pd), tsig(k_t, k_r, pd) \rangle$. In the discussion, ma_1 shall be called the *abort request for pd*, mr_1 the *resolve request for pd*, *ab_tok* the *abort_token for pd* and *res_cn* the *resolution for pd*. We first state some important properties of database of T that shall be used in the proofs of fairness and effectiveness.

3.2.1 Database Properties

The proof of fairness of the protocol depends on persistence and consistency of database of T. The proof of these properties relies on the freshness of the globally unique identifier in the role generation rule RG and uses the following proposition:

Proposition 3.2.1. *If S is a state reachable from S_1, then S contains exactly one of the following:*

$T_0(pd)$, $T_{ab}(pd, aborted, ab\_tok)$, $T_{or}(pd, resolved, res\_cn)$, $T_{rr}(pd, resolved, res\_cn)$, $T_{abf}(pd, aborted, ab\_tok)$, $T_{orf}(pd, resolved, res\_cn)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$.

Furthermore, there is at most one occurrence of the above facts in S_1.

In order to state the properties of persistence and consistency formally, we need the following definition:

Definition 3.2.2. Let pd be the protocol instance $pd = <m, n, k_o, k_r, k_t>$ that gets generated by the transition from S_0 to S_1. For states S reachable from S_1, we say

1. T has no entry for *pd* if $T_0(pd) \in S$.
2. T has an abort for *pd* if $T_{ab}(pd, aborted, ab\_tok)$ or $T_{abf}(pd, aborted, ab\_tok) \in S$.
3. T has a resolution for *pd* if S contains one of the following: $T_{or}(pd, resolved, res\_cn)$, $T_{rr}(pd, resolved, res\_cn)$, $T_{orf}(pd, resolved, res\_cn)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$.
4. T has answered a request for *pd* on the *O-T* channel if S contains one of the following: $T_{or}(pd, resolved, res\_cn)$, $T_{orf}(pd, resolved, res\_cn)$ $T_{ab}(pd, aborted, ab\_tok)$, $T_{abf}(pd, aborted, ab\_tok)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$.
5. T has answered a request for *pd* on the *R-T* channel if S contains one of the following: $T_{abf}(pd, aborted, ab\_tok)$, $T_{orf}(pd, resolved, res\_cn)$, $T_{rr}(pd, resolved, res\_cn)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$.

Now, we are ready to state the database properties of persistence and consistency which will help us in proving fairness for honest signers. The proofs of these properties (see section 3.4) depend on the uniqueness of the globally unique identifier n and are carried out by induction on the number of steps it takes to reach S from S_1. We have:

Lemma 3.2.3. ***Database Persistence*** *: For all states S reachable from S_1, one of the following is true:*

1) T has no entry for pd.

2) T has an abort for pd.

3) T has a resolution for pd.

If T has an abort for pd, then for all states S' reachable from S, T has an abort for pd in S'. If T has a resolution for pd, then for all states S' reachable from S, T has a resolution for pd in S'.

Lemma 3.2.4. ***Database Consistency*** *: For all states S reachable from S_1, the following are true:*

1) If T has no entry for pd in S, then T does not have an abort or a resolution for pd in S.

2) If T has an abort for pd in S, then T does not have a resolution for pd in S.

3) If T has a resolution for pd in S, then T does not have an abort for pd in S.

The following lemma which depends on database consistency and persistence shall be used to prove fairness. The detailed proof is carried out in section 3.4.

Lemma 3.2.5. *For all states S reachable from S_1, an abort_token for pd is in the view of the intruder at S only if T has an abort_token for pd. A resolution for pd is in the view of the intruder at S only if T has a resolution for pd.*

3.2.2 Effectiveness and fairness for honest O

We shall now state and prove effectiveness and fairness in our formalism if the signer playing the role O is honest. Assume that in the protocol instance generated by the transition from S_0 to S_1, O is honest and R strongly dishonest. By $O_i(pd, -)$ we mean O in its i−th state with first argument as pd.

We have the following proposition:

Proposition 3.2.6. *If S is a state reachable from S_1, then S contains exactly one of the following:*

$O_0(pd)$, $O_1(pd, me_1)$, $O_2(pd, me_1, me_2)$, $O_3(pd, me_1, me_2, me_3)$,

$O_{com}(pd, me_1, me_2, me_3, me_4)$, $O_{ab?}(pd, me_1, ab\_req)$,

$O_{res?}(pd, me_1, me_2, me_3, res\_req)$, $O_{ab1}(pd, me_1, ab\_req, ab\_tok)$,

$O_{ab2}(pd, me_1, me_2, me_3, ab\_req, ab\_tok)$, $O_{res1}(pd, me_1, ab\_req, res\_cn)$,

$O_{res2}(pd, me_1, me_2, me_3, res\_req, res\_cn)$.

Furthermore, there is exactly one occurrence of the fact $O_i(pd, -)$ for each $i \in \{0, 1, 2, 3, ab?, ab1, ab2, res?, res1, res2, com\}$.

The uniqueness of O_i is a result of the freshness of the unique identifier n and the proof is by induction on the length of trace from S_1 to S. The proof is given in section 3.4. In the light of the above proposition, we have the following definition:

Definition 3.2.7. For all states S reachable from S_1,

1. An honest O has completed pd in S if $O_i(pd, -) \in S$ for $i \in \{ab1, ab2, res1,$

$res2, com\}$.

2. An honest O has an abort_token for pd in S if $O_i(pd, -) \in S$ for $i \in \{ab1, ab2\}$.

3. An honest O has R's signature for pd in S if $O_i(pd, -) \in S$ for $i \in \{res1, res2, com\}$.

In order to state fairness for honest O, we need to define when R has O's signature on pd. Since R is strongly dishonest, we identify R with the intruder:

Definition 3.2.8. Let S be reachable from S_1. A strongly dishonest R is said to have O's signature for pd in S if $sig(k_o, pd)$ or $tsig(k_t, k_o, pd)$ is in the view of the intruder at S. It is said to have the abort_token for pd, if $sig(k_t, sig(k_o, <abort, pd>))$ is in the view of the intruder at S.

Now we define, effectiveness and fairness for honest O. We break the definition in [28] into three definitions. The definitions for an honest R will be stated similarly.

Definition 3.2.9. Completeness for honest O. There is a state S reachable from S_1, such that O has completed pd in S and has R's signature on pd.

Intutively, completeness says that an honest O may be able to exchnage signatures with R. A protocol is said to be effective for honest O if at any stage of the protocol, honest O may complete the protocol with the help of T:

Definition 3.2.10. Effectiveness for honest O. For all states S reachable from S_1, there is a state S' reachable from S, using only the rules in $\mathbf{O} \cup \mathbf{T}$, such that

a) O has completed *pd* in S', and

b) O has either R's signature for *pd* or an abort_token for *pd* in S'.

Because of the nondeterminism in our system, we cannot prove the stronger version stated in [28]. Now, we are ready to state fairness of honest O. A protocol is said to be fair for honest O, if honest O can get R's signature whenever R gets O's signature:

Definition 3.2.11. Fairness for honest O. For all states S reachable from S_1,

1. If a strongly dishonest R has O's signature for *pd* in S, then there exists S' reachable from S using transitions in $\mathbf{O} \cup \mathbf{T}$ such that an honest O has R's signature for *pd* in S'.

2. If an honest O has an abort_token for *pd* in S, then for all S' reachable from S, a strongly dishonest R does not have O's signature for *pd* in S'.

Please note that it can be easily shown that once O has an abort_token for *pd*, it can never have R's signature on *pd*. In light of this fact, the condition 2 in the defintion of 1 is a consequence of condition 1. However, we shall keep these two conditions separate as were kept in the [28]. Now we shall discuss effectiveness and fairness is somewhat greater detail.

Effectiveness. The proof of effectiveness for honest O depends on the following two lemmas. The detailed proofs of these two lemmas are given in section 3.4. The proofs of these lemmas depend on uniqueness of the global identifier n and on the special properties of the O-T channel.

Lemma 3.2.12. *If in a state S reachable from S_1, O is in a state in which it has requested for an abort_token from T for pd and is waiting for a reply,* i.e., *$O_{ab?}(pd, -) \in S$, then*

<u>either</u> T has answered a request for pd on the O-T channel, and an abort_token or a resolution for pd is on the O-T channel,

<u>or</u> T has yet to answer a request for pd on the O-T channel, and the abort request for pd is on the O-T channel.

Lemma 3.2.13. *If in a state S reachable from S_1, O is in a state in which it has requested for a resolution from T for pd and is waiting for a reply,* i.e., *$O_{res?}(pd, -) \in S$, then*

<u>either</u> T has answered a request for pd on the O-T channel, and an abort_token or a resolution is on the O-T channel.

<u>or</u> T has yet to answer a request for pd on the O-T channel, and the request is on the O-T channel.

Now we are ready to prove effectiveness.

Theorem 3.2.14. *The protocol is effective for honest O,* i.e., *for all states S reachable from S_1, there exists S' reachable from S, using only the rules in* **O** *and* **T***, such that O has completed pd in S' and has either O's signature on pd or an abort token for pd in S'.*

Proof. Intuitively, at each point in the exchange subprotocol, O can always progress by either sending a message to R or contacting T with an abort or resolve request. T is honest and replies to the request. O reads the answer and reaches a final role state.

By proposition 3.2.6, for all reachable S, O is in exactly one of the 10 states. Hence the following cases arise:

1. $O_i(pd, -) \in S$ for $i \in \{ab1, ab2, res1, res2, com\}$: O has completed pd and has either an abort token or a valid contract for pd.

2. $O_{res?}(pd, -) \in S$: By lemma 3.2.13 either T has answered a request on the O-T channel and an abort_token or a resolution for pd is on the O-T channel in S, or T has yet to answer a request for pd on the O-T channel and a resolve request is on the O-T channel. In the former case, O reads the abort_token or the resolution and completes pd, and in the latter case T responds to the resolve request and then O completes pd.

3. $O_3(pd, -) \in S$: O requests T for a resolution. The result follows from case 2.

4. $O_2(pd, -) \in S$: O sends me_3 and the result follows from case 3.

5. $O_{ab?}(pd, -) \in S$: Similar to case 2 using lemma 3.2.12.

6. $O_1(pd, -) \in S$: O requests T for an abort_token and the result follows from case 5.

7. $O_0(pd, -) \in S$: O sends me_1 and the result follows from case 6.

□

Theorem 3.2.15. *The protocol is complete for honest O,* i.e., *there is a state S reachable from S_1, such that O has completed pd in S and has R's signature on pd.*

Proof. The result follows when both O and R follow the exchange subprotocol with no rules of intruder and T being used. □

Fairness. The proof of fairness for honest O depends on the following three lemmas. The detailed proofs of these lemmas are carried out in section 3.4

Lemma 3.2.16. *For all states S reachable from S_1,*

1. *If O has an abort_token for pd then T has an abort_token for pd. Furthermore, in all states S' reachable from S, a resolution for pd is not in the view of the intruder in S'.*

2. *If a resolution for pd is in the view of the intruder in S then for all states S' reachable from S, O does not have an abort_token.*

The proof is by induction and uses lemma 3.2.5, database consistency and persistence.

Lemma 3.2.17. *For all states S reachable from S_1,*

1. *If $O_i(pd, -)$ in S, where $i \in \{0, 1, 2, 3, res?\}$, then neither an abort request nor an abort_token for pd is in the view of the intruder at S.*

2. *S does not contain $O_{ab2}(pd, -) \notin S$,* i.e., *the only way O has an abort_token is if it is in the state O_{ab1}.*

The proof is by induction and uses the fact that the intruder does not learn the signing key of O.

Lemma 3.2.18. *For all reachable S,*

1. *If S contains, $O_i(pd, -)$ for $i \in \{0, 1, ab?, ab1, res1\}$, then O's signature on pd, $sig(k_o, pd)$, in not in the view of the intruder.*

2. *If O has an abort_token for pd then for all S' reachable from S, O's signature on pd, $sig(k_o, pd)$, is not in the view of the intruder. If O's signature on pd, $sig(k_o, pd)$, is in the view of the intruder then for all S' reachable from S, O does not have an abort_token in S'.*

The proof is by induction and uses the fact that the intruder does not learn the signing key of O.

Theorem 3.2.19. *The protocol is fair for honest O.*

Proof. If R has O's signature on *pd* is a state S, then by effectiveness for honest O, there is a state S' reachable from S using transitions in $\mathbf{O} \cup \mathbf{T}$ such that O has completed *pd* in S' and has either an abort_token for *pd* or R's signature on *pd* in S'. By lemma 3.2.16 and lemma 3.2.18, since R has O's signature for *pd* in S, we get O cannot have and abort_token for *pd* in S'. Hence, O must have R's signature on *pd* in S'.

If O has an abort_token for pd in S, then by lemma 3.2.16 and lemma 3.2.18, R cannot have O's signature on pd for any state S' reachable from S. □

The following is a corollary of the proof of fairness of honest O:

Corollary 3.2.20. *Effectiveness and fairness for honest O holds even if the intruder takes a bounded number of steps and the role generation rule is used a bounded number of times.*

3.2.3 Effectiveness and fairness for honest R

We shall now state and prove effectiveness and fairness in our formalism if the signer playing the role R is honest. Assume that in the protocol instance generated by the transition from S_0 to S_1, R is honest and O strongly dishonest. By $R_i(pd, -)$ we mean R in its i−th state with first argument as pd.

We have the following proposition:

Proposition 3.2.21. *If S is a state reachable from S_1, then S contains exactly one of the following:*
$R_0(pd)$, $R_{quit}(pd)$, $R_1(pd, me_1)$, $R_2(pd, me_1, me_2)$, $R_3(pd, me_1, me_2, me_3)$,
$R_{com}(pd, me_1, me_2, me_3, me_4)$, $R_{res?}(pd, me_1, me_2, res\_req)$,
$R_{ab}(pd, me_1, me_2, ab\_req, ab\_tok)$, $R_{res}(pd, me_1, me_2, res\_req, res\_cn)$.
Furthermore, there is exactly one occurrence of the fact $R_i(pd, -)$ for each
$i \in \{0, 1, 2, 3, quit, ab?, res?, ab, res, com\}$.

The uniqueness of R_i is a result of the freshness of the nonce n and the proof is by induction on the length of trace from S_1 to S. The proof is given in section 3.4. In the light of the above proposition, we have the following definition:

Definition 3.2.22. For all states S reachable from S_1,

1. An honest R has completed *pd* in S if $R_i(pd, -) \in S$ for $i \in \{quit, res, ab, com\}$.
2. An honest R has an abort_token for *pd* in S if $R_{ab}(pd, -) \in S$.
3. An honest R has quit *pd* in S if $R_{ab}(pd, -) \in S$.
4. An honest R has O's signature for *pd* in S if $R_i(pd, -) \in S$ for $i \in \{res, com\}$.

Now we define when O has R's signature on *pd*. Since O is strongly dishonest, we identify O with the intruder:

Definition 3.2.23. Let S be reachable from S_1. A strongly dishonest O is said to have R's signature on *pd* in S if $sig(k_r, pd)$ or $tsig(k_t, k_r, pd)$ is in the view of the intruder at S. It is said to have the abort_token for *pd*, if $sig(k_t, sig(k_o, <abort, pd>))$, in the view of the intruder at S.

Now we define fairness and effectiveness for honest R.

Definition 3.2.24. Completeness for honest R. There is a state S reachable from S_1, such that R has completed *pd* in S and has O's signature on *pd*.

Intuitively, completeness says that an honest R may be able to exchange signatures with O. A protocol is said to be effective for honest R if at any stage of the protocol, honest R may complete the protocol with the help of T:

Definition 3.2.25. Effectiveness for honest R. For all states S reachable from S_1, there is a state S' reachable from S, using only the rules in $\mathbf{R} \cup \mathbf{T}$, such that
a) R has completed *pd* in S', and
b) R has O's signature for *pd*, or has quit *pd*, or has an abort token for *pd* in S'.

Now, we are ready to state fairness of honest R. A protocol is said to be fair for honest R, if honest R can get O's signature whenever O gets R's signature:

Definition 3.2.26. Fairness for honest R. For all states S reachable from S_1,

1. If a strongly dishonest O has R's signature for *pd* in S, then there exists S' reachable from S such that an honest R has O's signature for *pd* in S'.

2. If an honest R has quit *pd* or has an abort_token for *pd* in S, then for all S' reachable from S, a strongly dishonest O does not have R's signature for *pd* in S'.

Once again condition 2 can be shown to be a consequence of condition 1. Now we shall discuss effectiveness and fairness is somewhat greater detail.

Effectiveness. The proof of effectiveness for honest R depends on the following lemmas. The detailed proof of these this lemma is given in section 3.4. The proof of this lemma depends on freshness of the global identifier n and on the special properties of the R-T channel.

Lemma 3.2.27. *If in a state S reachable from S_1, R is in a state in which it has requested for a resolution from T for pd and is waiting for a reply,* i.e., $R_{res?}(pd-) \in S$*, then*

<u>either</u> T has answered a request for pd on the R-T channel and an abort_token or a resolution for pd is on the R-T channel.

<u>or</u> T has yet to answer a request for pd on the R-T channel and the resolve request is on the R-T channel.

Now we are ready to prove effectiveness.

Theorem 3.2.28. *The protocol is effective for honest R,* i.e., *for all states S reachable from S_1, there exists S' reachable from S, using only the rules in $\mathbf{R} \cup \mathbf{T}$, such that R has completed pd in S' and has either quit pd or has obtained either O's signature on pd or an abort_token for pd in S'.*

Proof. Intuitively, at each point in the exchange subprotocol, R can always progress by either sending a message to R or contacting T with an abort or resolve request. T is honest and replies to the request. R reads the answer and reaches a final role state.

By proposition 3.2.21, for all reachable S, R is in exactly one of the 10 states. Hence the following cases arise:

1. $R_i(pd, -) \in S$ for $i \in \{ab, res, quit, com\}$: R has completed *pd* and has either an abort token or O's signature for *pd*.

2. $R_{res?}(pd, -) \in S$: By lemma 3.2.27 either T has answered a request on the R-T channel and an abort_token or a resolution for *pd* is on the R-T channel in S, or T has yet to answer a request for *pd* on the R-T channel and a resolve request is on the R-T channel. In the former case, R reads the abort_token or the resolution and completes *pd*, and in the latter case T responds to the resolve request and then R completes *pd*.

3. $R_3(pd, -) \in S$: R sends O's its signature on *pd*. The result follows from case 2.

4. $R_2(pd, -) \in S$: R asks T for a resolve request and the result follows from case 2.

5. $R_1(pd, -) \in S$: R sends O the message me_2 and the result follows from case 5.

6. $R_0(pd, -) \in S$: R quits and the result follows from case 1.

$\square$

Theorem 3.2.29. *The protocol is complete for honest R,* i.e., *there is a state S reachable from S_1, such that R has completed pd in S and has O's signature on pd.*

Proof. The result follows when both O and R follow the exchange subprotocol with no rules of intruder and T being used. $\square$

Fairness The proof of fairness depends on the following three lemmas. The detailed proofs of these lemmas are carried out in 3.4.

Lemma 3.2.30. *For all states S reachable from S_1,*

1. *If S contains $R_i(pd, -)$ where $i \in \{0, quit\}$, then neither $PCS(k_r, pd, k_o, k_t)$, R's PCS on pd, nor a resolution for pd is in the view of the intruder. If R has quit pd, then for all states S' reachable from S, a resolution is not in the view of the intruder at S'.*

2. *If a resolution for pd is in the view of the intruder in S, then for all states S' reachable from S, R has not quit pd in S'.*

Lemma 3.2.31. *For all states S reachable from S_1,*

1. *If R has an abort_token for pd then T has an abort_token for pd. Furthermore, in all states S' reachable from S, a resolution for pd is not in the view of the intruder in S'.*

2. *If a resolution for pd is in the view of the intruder in S then for all states S' reachable from S, R does not have an abort_token for pd.*

Lemma 3.2.32. *For all states S reachable from S_1,*

1. *If S contains $R_i(pd, -)$ where $i \in \{0, quit, 1, 2, 3, ab?, ab, res\}$, then R's signature on pd, $sig(k_r, pd)$, is in not in the view of the intruder.*

2. *If R has quit pd or R has an abort_token for pd then for all states S' reachable from S, $sig(k_r, pd)$ in not in the view of the intruder. If $sig(k_r, pd)$ is in the view of the intruder then for all states S' reachable from S, R has neither quit pd nor has an abort_token for pd in S'.*

Theorem 3.2.33. *The protocol is fair for honest R.*

Proof. If O has R's signature on *pd* in S, then by effectiveness there is a state S' reachable from S such that R has completed *pd*. Now, since R has O'signature on *pd*, we get by lemma 3.2.30 and lemma 3.2.32, R has not quit *pd* in S'. Also by lemma 3.2.31 and lemma 3.2.32 R does not have an abort_token for *pd* in S'. Therefore, since R has completed *pd* in S', we R has O's signature in S'.

If R has quit *pd* in S, then by lemma 3.2.30, lemma 3.2.31 and lemma 3.2.32, O does not have R's signature on *pd* in any state S' reachable from S. □

Once again, we have the corollary:

Corollary 3.2.34. *Effectiveness and fairness for honest R holds even if the intruder takes a bounded number of steps and the role generation rule is used a bounded number of times.*

3.3 Balance

It is a consequence of fairness, that if a dishonest signer gets the honest signer's signature, then the honest signer will get the signature of the dishonest signer. However, even if the protocol is fair, the dishonest signer may still be able to decide if the signatures are exchanged or not. In this Section, we shall study a dishonest signer's ability to control the outcome of the protocol provided the other signer is honest. For our analysis, we shall assume that one of the signers is honest, and the other one dishonest. For the sake of simplicity, we shall consider a single isolated run of the protocol. However, our definitions and proofs can be easily lifted to multiple concurrent runs. We start by giving a few preliminaries.

Recall that Σ is the initial finite set of facts. Assume that in Σ, two signers, O and R whose private/public key pairs are (k_{os}, k_o), and (k_{rs}, k_r), agree to exchange signatures with pre-agreed text m, unique identifier n, and a trusted third party T, whose private/public key pair is (k_{ts}, k_t). This they do by the use of role generation rule RG. Let the resulting state be S_1. Let $pd = <m, n, k_o, k_r, k_t>$. Let A be one of the signers, O or R, and let B be the other signer. For the rest of this section, we shall assume that A is honest and B strongly dishonest. If A is O, let $\mathbf{A}$ be $\mathbf{O}$ and $\mathbf{B}$ be $\mathbf{R}$. If A is R, let $\mathbf{A}$ be $\mathbf{R}$ and $\mathbf{B}$ be $\mathbf{O}$.

For the rest of the section, by a reachable state S, we mean a state reachable from S_1 without the use of the role generation rule, RG. Note that since we do not allow the use of role generation, we are considering only a single instance of the protocol in isolation and not a concurrent run of several instances.

It can be easily seen that if the intruder takes only a bounded number of steps, and no role generation rules are used, then the number of traces from S_1 is finite. Given a finite trace, $S^{(1)}, \ldots, S^{(n)}$, from a state $S^{(1)}$, construct a labeled chain: its nodes are labeled by $S^{(1)}, \ldots, S^{(n)}$ respectively, and its edges are labeled by a three 3 tuple. The first argument of this tuple is a state transition rule, second argument is a ground substitution and the third argument is a theory in the set $\{\mathbf{O}, \mathbf{R}, \mathbf{T}, \mathbf{I}\}$. It must be the case that if $< t, \sigma, \mathbf{Q} >$ labels the edge connecting nodes labeled by $S^{(i)}$ and $S^{(i+1)}$, then the application of t using σ transforms $S^{(i)}$ to $S^{(i+1)}$, and $t \in \mathbf{Q}$. For simplicity sake, we shall sometimes say that the edge is labeled by a transition in $\mathbf{Q}$ if $t \in \mathbf{Q}$. We call such a chain, a *labeled trace*. Given a state S reachable from S_1, let the *continuation tree from* S be the tree of all the possible labeled traces starting from the state S.

The continuation tree from S may be thought of as the tree of all possible plays in which the signers, the trusted third party and the intruder are participants. A strongly dishonest signer may consider some of these plays favorable and some others unfavorable to its goals. Now, a strongly dishonest signer cannot control the actions of T and the actions of the other signer. However, it can control some or all of its actions. Also, since the strongly dishonest R is in coalition with the intruder, it may control the actions of the intruder. Keeping this in mind, we have the following definition:

Definition 3.3.1. Let *ctr* be the continuation tree from S. An edge of *ctr* is said to be a *removable edge* or *under B's control* if it is labeled by a rule in $\mathbf{R} \cup \mathbf{I}$.

Now, we are ready to define B's ability to control the outcome of the protocol. As a consequence of fairness, there are two possible outcomes: either both signers will get each other's signatures or none will. The following definition is motivated by game strategies.

Definition 3.3.2. Let S be a reachable state and assume that the intruder takes a bounded number of steps. Let *ctr* be the continuation tree from S. Let E be a subset of removable edges in *ctr*. Define $ctr \backslash E$, *a strategy*, to be the tree obtained from *ctr* by removing the edges in E and all of their descendants.

1. *At S, B has the power to abort* if there is a selection E such that the leaf nodes of $ctr \backslash E$ are labeled by states in which A does not have B's signature on *pd*. We call $ctr \backslash E$ an *abort-tree.*

2. *At S, B has the power to complete* if there is a selection E such that the leaf nodes of $ctr \backslash E$ are labeled by a states in which B has A's signature on *pd.* We call $ctr \backslash E$ a *resolve-tree.*

3. B has the *advantage over an honest A at S* if at S, B has both the power to abort and the power to complete.

In the above definition, E represents the actions that B considers unfavorable and $ctr \backslash E$ represents the continuations that B prefers. If, by choosing not to do certain actions, B may prevent A from getting its signature then we say that B has the power the abort. In this case B may force the exchange not to happen. If, by choosing not to do certain actions, B may ensure exchange of signatures, then we say B has the power to complete. If B can force both outcomes, then B is said to have the advantage. If B does not enjoy the advantage, we say that the protocol is balanced:

Definition 3.3.3. The protocol is said to be *balanced for honest* A if for all reachable S, and for all bounds on the number of steps an intruder takes, B does not have have an advantage over honest A at S.

Note that for the protocol not to be balanced, all we need is to show that B has the power to abort and the power to complete for one bound on the number of steps that the intruder could take (the intruder is in coalition with B).

3.3.1 Recursive characterization of balance

We shall now give a recursive characterization of balance. We shall use this recursive characterization to prove balance for honest signers for the revised protocol.

Definition 3.3.4. Let S be a reachable state and assume that the intruder takes only finitely many steps. Let *ctr* be the continuation tree of S. Let N be a node in *ctr* and X a set of nodes that are children of N such that any edge between N and a member of X is a removable edge. Define N_X to be the set of children of N that are either in X or connected to N by a non-removable edge. N is an *abort-power* *node* if

<u>either</u> there is an X such that N_X is nonempty and each node in N_X is an abort-power node,

<u>or</u> there is an X such that N_X is empty and A does not B's signature on *pd* in the state labeling N.

In the above definition X is the set of actions that B considers favorable at the node N if B is trying to prevent the exchange of signatures. Note that in the definition above, X may be empty (choosing not to do anything is a valid strategy). Also note that in the above definition, we consider the two cases: N_X being non-empty, and N_X being empty separately. This is to rule out cases such as in which we are at a leaf node and A has B's signature. If we do not consider the empty N_X case, then the first condition will be vacuously true. Also since *ctr* is a finite tree, the above recursive definition makes sense. Now we give another recursive definition:

Definition 3.3.5. Let S be a reachable state and assume that the intruder takes only finitely many steps. Let *ctr* be the continuation tree of S. Let N be a node in *ctr* and X a set of nodes that are children of N such that any edge between N and a member of X is a removable edge. Define N_X to be the set of children of N that are either in X or connected to N by a non-removable edge. N is an *resolve-power node* if

<u>either</u> there is an X such that N_X is nonempty and each node in N_X is an resolve-

power node,

or there is an X such that N_X is empty and A has B's signature on *pd* in the state labeling N.

For the rest of the section we assume, that S is a reachable state, the intruder takes a bounded number of steps and *ctr* is the continuation tree of S with respect to this bound. We have the following connection between the power to abort and abort-power nodes:

Lemma 3.3.6. *Let ctr be the continuation tree from S. At S, B has the power to abort if and only if the root of ctr is an abort-power node. Also at S, B has the power to complete if and only if the root of ctr is a resolve-power node.*

The proof is by induction on the height of a node in the tree *ctr*. We show that if at S, B has the power to abort at S then all the nodes in an abort-tree are abort-power nodes. For the other direction, we show how to choose a set of edges in *ctr*, inductively such that $ctr\backslash E$ is an abort-power tree. The details are in section 3.4.

Please note that because of fairness (corollaries 3.2.20 and 3.2.34), once an honest A gets an abort_token for *pd* or quits for *pd*, then B cannot obtain A' signature. Keeping this in mind, we state the following proposition. A detailed proof of this proposition is given in section 3.4.

Proposition 3.3.7. *We have*

1) If a node N in ctr is labeled by a state in which either A has an abort_token for *pd or A has quit pd, then N is not a resolve-power node.*

2) If a node N in ctr is labeled by a state in which A has B's signature on pd, then N is not an abort-power node.

3) Let N and N' be nodes in ctr such that N' is a child of N and the edge between N and N' is non-removable. If N' is not an abort-power node, then N is not an abort-power node. If N' is not a resolve-power node, then N is not a resolve power node.

3.3.2 Balance for honest O

We shall now show that the protocol is balanced if the honest signer A is O. In this case, we have B is R, **A** is **O**, and **B** is **R**. The proof relies on the lemmas 3.2.12 and 3.2.13 that we used in proving effectiveness for honest O (section 3.2.2). We shall use the recursive characterization of balance (definitions 3.3.4 and 3.3.5) in order to show balance for honest O. We start by proving a couple of propositions.

Proposition 3.3.8. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $O_{ab?}(pd, -)$,* i.e., *if O has requested T to abort and is waiting for a reply, and if T has answered a request on the O-T channel in S', then*

either N is not an abort-power node, or N is not a resolve-power node.

Proof. By proposition 3.2.1 and lemma 3.2.12, either an abort_token for *pd* is on the *O-T* Channel or a resolution for *pd* is on the *O-T* channel in S'. If the abort_token is on the O-T channel, then O can read it by the use of rule O_{ab1} and go to a state in which O has an abort_token. In this case N has child N' such that

a) The edge between N and N' is labeled by a rule in **O**. Clearly this edge is non-removable.

b) N' is labeled by a state in which O has an abort_token for *pd*. By proposition 3.3.7, N' is not a resolve-power node.

By proposition 3.3.7, N is not a resolve-power node. Similarly, if a resolution from T is on the O-T channel, we have N is not an abort-power node. □

Proposition 3.3.9. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $O_{ab?}(pd, -)$,* i.e., *if O has requested T to abort and is waiting for a reply, and if T has yet to answer a request on the O-T channel in S', then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

Proof. By proposition 3.2.1 and lemma 3.2.12, the abort request for *pd* is on the O-T Channel in S'. Since T has yet not answered a request on the O-T channel in S', it can answer it and we obtain a in which T has answered a request on the O-T channel. We get N has a child N' such that

a) The edge between N and N' is labeled by a rule in **T**. Clearly this edge is non-removable.

b) N' is labeled by a state that contains $O_{ab?}(pd-)$ and in which T has answered a request on the O-T channel.

By proposition 3.3.8 either N' is not an abort-power node or N' is not a resolve power node. By proposition 3.3.7, if N' is not an abort-power node, then N is not an abort-power node. If N' is not a resolve-power node, then N is not a resolve-power node. $\square$

We can combine the above two propositions and proposition 3.2.12 to prove the following:

Lemma 3.3.10. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $O_{ab?}(pd, -)$,* i.e., *if O has requested T to abort and is waiting for a reply, then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

Proof. By lemma 3.2.12, T has either answered a request for *pd* on the O-T channel in S', or T is yet to answer a request for *pd* on the O-T channel in S'. The result now follows by propositions 3.3.8 and 3.3.9. $\square$

We shall also need the following lemma:

Lemma 3.3.11. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $O_{res?}(pd, -)$,* i.e., *if O has requested T to resolve pd and is waiting for a reply, then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

The proof is similar to lemma 3.3.10 and uses lemma 3.2.13. The details are in section 3.4. Now, we are ready to show that the protocol is balanced for honest O.

Lemma 3.3.12. *Let N be a node in ctr. Then* <u>*either*</u> *N is not an abort-power node,* <u>*or*</u> *N is not a resolve-power node.*

Proof. : Let N be labeled by S'. Since S' is a reachable state, by proposition 3.2.6, we have O must be in one of its ten states in S'. So the following cases arise:

1. $O_{ab?}(pd, -) \in S'$: The result follows immediately from lemma 3.3.10.

2. $O_1(pd, -) \in S'$: O can request an abort_token from T by the use of rule $O_{ab?}$ and go to a state S", such that $O_{ab?}(pd, -) \in S"$. Then N is connected to a non-removable edge to a node N' labeled by $S"$. By case 1, either N is not an abort-power node or N is not a resolve-power node. The result then follows from proposition 3.3.7.

3. $O_0(pd, -) \in S'$: O can send me_1 by the use rule O_1 a go to a state S", such that $O_1(pd, -) \in S"$. The result then follows by case 2 and proposition 3.3.7 .

4. $O_{res?}(pd, -) \in S'$: The result follows from lemma 3.3.11.

5. $O_3(pd, -) \in S'$: O can request a resolution from T by the use of rule $O_{res?}$ and go to a state $S"$, such that $O_{res?}(pd, -) \in S"$. The result follows by case 4 and proposition 3.3.7.

6. $O_2(pd, -) \in S'$: O can send me_2 by the use of rule O_3 and go to a state $S"$, such that $O_3(pd, -) \in S"$. The result follows by case 5 and proposition 3.3.7.

7. $O_{ab1}(pd, -) \in S'$ or $O_{ab2}(pd, -) \in S'$: O has an abort_token and by proposition 3.3.7, N is not a resolve-power node.

8. $O_{res1}(pd, -) \in S'$ or $O_{res2}(pd, -) \in S'$ or $O_{com}(pd, -) \in S'$: O has R' signature on pd and by proposition 3.3.7, N is not an abort-power node.

□

Theorem 3.3.13. *The protocol is balanced for honest O.*

Proof. The theorem follows immediately from lemma 3.3.6 and lemma 3.3.12. □

The key ingredients in the proof of lemma 3.3.12 are: lemma 3.2.12, lemma 3.2.13 and the ability of an honest O to contact T non-deterministically at any step of the protocol. These are the same ingredients that go into proving effectiveness and fairness for honest O. This is not an accident and we shall show in chapter 6, that in case of single runs a signature-exchange protocol is balanced for honest signers if it is fair and effective. Here, we adopted the approach of giving a recursive characterization mainly for two reasons which we outline here.

This proof clearly highlights that a key ingredient in proving balance is O's ability to contact T non-deterministically at each stage of the protocol. However, as we argue in chapter 1, a real-world O will not exercise these options non-deterministically. We shall refine our model to reflect the real-world behavior in chapter 4.

This recursive characterization can be used for all protocols, even protocols that are not effective. This recursive characterization also suggests an algorithm for automating verification for balance. For example, in order to check whether a signer B has power to resolve against an honest A, we first construct the continuation tree from S, assuming the intruder makes a finite number of steps. Let the height of this tree be i. Now look at the nodes at level i: all nodes in which B has A's signature are marked as resolve-power nodes and every other node is marked as non-resolve-power nodes. Now suppose that all nodes up to level j have been marked. Now, consider a node N at level $j-1$. Two cases arise:

a) N has a child connected by non-removable edge. If all children of N connected by non-removable edges are resolve-power nodes, then N is a resolve-power node (In this case X in definition 3.3.4 is empty.) If any such child is not a resolve-power node, then N is not resolve-power node.

b) N does have a child connected by non-removable edge. If B has A's signature in N, then N is a resolve-power node. Otherwise, N is a resolve-power node if and only if there is a removable edge connecting N to a resolve-power node.

B has power to resolve if the root is marked as resolve-power node (proposition 3.3.6).

The above definitions and proofs of balance for honest O can be extended to multiple concurrent runs of the protocol:

1) Instead of initial set of facts, Σ, we start with an arbitrary reachable state.

2) For the construction of a finite height continuation tree, along with the number of steps that the intruder steps, we also put a bound the number of times the role generation rule can be used.

3) Each edge of the continuation tree is also labeled with with the key of the principal involved.

4) Any edge that is labeled by R's key or a key known to the intruder is also a removable edge.

5) The protocol is said to be balanced for honest O, if for all reachable states S, and for all bounds on the number of steps that the intruder takes and the number of times the role generation rule is used, at S, R does not have both the power to abort and the power to complete.

3.3.3 Balance for honest R

We shall now show that the protocol is balanced if the honest signer A is R. In this case, we have B is O, **A** is **R**, and **B** is **O**. The proof relies on the lemma 3.2.27 we used in proving effectiveness for honest R (section 3.2.3). We shall use the recursive characterization of balance (definitions 3.3.4 and 3.3.5) in order to show balance for honest R. The proof of balance depends on the following lemma:

Lemma 3.3.14. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $R_{res?}(pd, -)$,* i.e., *if R has requested T to resolve pd and is waiting for a reply, then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

The proof is similar to lemma 3.3.10 and uses lemma 3.2.27. The details are in section 3.4. Now, we are ready to show that the protocol is balanced for honest R. First we show

Lemma 3.3.15. *Let N be a node in ctr. Then either N is not an abort-power node or N is not a resolve-power node.*

Proof. : Let N be labeled S'. Since S' is a reachable state, by proposition 3.2.21, we have R must be in one of its nine states in S'. So the following cases arise:

1. $R_{quit}(pd, -) \in S'$: By proposition 3.3.7, N is not a resolve-power node.

2. $R_0(pd, -) \in S'$: R can quit by the use of rule R_{quit} and the result follows from part 1 and proposition 3.3.7

3. $R_{res?}(pd, -) \in S'$: The result follows from lemma 3.3.14.

4. $R_2(pd, -) \in S'$: R can ask for a resolution by the use of rule $R_{res?}$ and the result follows from case 3 and proposition 3.3.7.

5. $R_1(pd, -) \in S'$: R can send me_2 by the use of rule R_2 and the result then follows from case 4 and proposition 3.3.7.

6. $R_{ab}(pd,_)) \in S'$: R has an abort_token for pd and by proposition 3.3.7, N is not a resolve-power node.

7. $R_{res}(pd, -) \in S'$ or $R_{com}(pd, -) \in S'$ or $R_3(pd, -) \in S'$: R has O's signature on pd and by proposition 3.3.7, N is not an abort-power node.

□

Theorem 3.3.16. *The protocol is balanced for honest R.*

Proof. The theorem follows immediately from lemma 3.3.6 and lemma 3.3.15. □

The key ingredients in the proof of lemma 3.3.12, are: lemma 3.2.27, and the ability of an honest R to quit or contact T non-deterministically at any step of the protocol. These are the same ingredients that go into proving effectiveness and fairness for honest R. However, once again as we argued in chapter 1, a real-world R will not exercise these options non-deterministically, and we shall refine our model to reflect the real-world behavior in chapter 4.

Once again the definition and proof can be extended for concurrent runs.

3.4 Proofs

3.4.1 Proofs of facts of the view of the intruder

We first state few facts about the definitions we gave in section 3.1.

Proposition 3.4.1. *Let k, k_1, k_2, k_3 be constants of the sort public_key and k^{-1}, k_1^{-1}, k_2^{-1}, k_3^{-1} be the corresponding private keys. Let m be a term of the sort mssg. For any set of terms, T, T_1 and T_2, we have*

1. *If $T_1 \subseteq analz(T_2)$ then $analz(T_1) \subseteq analz(T_2)$. If $T_1 \subseteq synth(T_2)$ then $synth(T_1) \subseteq synth(T_2)$.*

2. *If $T_1 \subseteq T_2$ then $analz(T_1) \subseteq analz(T_2)$ and $synth(T_1) \subseteq synth(T_2)$.*

3. *$analz(T_1) \bigcup analz(T_2) = analz(T_1 \bigcup T_2)$.*

4. *If $k \in synth(T)$ then $k \in T$. If $k^{-1} \in synth(T)$ then $k^{-1} \in T$.*

5. *If $PCS(k_1, m, k_2, k_3) \in synth(analz(T))$ then either $PCS(k_1, m, k_2, k_3) \in analz(T)$ or $k_1^{-1} \in analz(T)$.*

6. *If $sig(k, m) \in synth(analz(T))$ then either $sig(k, m) \in analz(T)$ or $k^{-1} \in analz(T)$.*

7. *If $tsig(k, k_1, m) \in synth(analz(T))$ then either $tsig(k, k_1, m) \in analz(T)$ or $k^{-1} \in analz(T)$.*

Proof. 1. We are given $T_1 \subseteq analz(T_2)$. By definition of $analz$, $analz(T_1)$ is the smallest superset, X, of T_1 such that

$u \in X$ and $v \in X$ if $\langle u, v\rangle \in X$,

$u \in X$ if $sig(k, u) \in X$,

$u \in X$ if $PCS(k, u, k_1, k_t) \in X$,

$u \in X$ if $FakeSign(k, u, k_1, k_t) \in X$, and

$u \in X$ if $tsig(k, k_1, u) \in X$.

By definition of $analz(T_2)$, $analz(T_2)$ is one such set. Therefore we get $analz(T_1) \subseteq analz(T_2)$. Similarly, $synth(T_1) \subseteq synth(T_2)$.

2. We have by definition of $analz$, $T_2 \subseteq analz(T_2)$ and $T_2 \subseteq synth(T_2)$. Hence if $T_1 \subseteq T_2$, then $T_1 \subseteq analz(T_2)$ and $T_1 \subseteq synth(T_2)$. Hence by part 1, $analz(T_1) \subseteq analz(T_2)$ and $synth(T_1) \subseteq synth(T_2)$.

3. We have $analz(T_1) \subseteq analz(T_1 \bigcup T_2)$ and $analz(T_2) \subseteq analz(T_1 \bigcup T_2)$ (by part 2). Hence, $analz(T_1) \bigcup analz(T_2) \subseteq analz(T_1 \bigcup T_2)$.

   Now, we have $T_1 \subseteq analz(T_1)$ and $T_2 \subseteq analz(T_2)$. Hence, we have by definition $T_1 \bigcup T_2 \subseteq analz(T_1) \bigcup analz(T_2)$.

   Now, if $\langle u, v \rangle \in analz(T_1) \bigcup analz(T_2)$, then either $\langle u, v \rangle \in analz(T_1)$ or $\langle u, v \rangle \in analz(T_2)$. If $\langle u, v \rangle \in analz(T_1)$, then by definition of $analz$, $u \in analz(T_1)$ and $v \in analz(T_1)$. Since $analz(T_1) \subseteq analz(T_1) \bigcup analz(T_2)$, we get $u \in analz(T_1) \bigcup analz(T_2)$ and $v \in analz(T_1) \bigcup analz(T_2)$. Similarly if $\langle u, v \rangle \in analz(T_2)$, we get $u \in analz(T_1) \bigcup analz(T_2)$ and $v \in analz(T_1) \bigcup analz(T_2)$.

   Similarly, we can show that $analz(T_1) \bigcup analz(T_2)$ is a set X such that

   $u \in X$ if $sig(k, u) \in X$, $u \in X$ if $PCS(k, u, k_1, k_t) \in X$,

   $u \in X$ if $FakeSign(k, u, k_1, k_t) \in X$, and $u \in X$ if $tsig(k, k_1, u) \in X$.

   Therefore by definition of $analz$, $analz(T_1 \bigcup T_2) \subseteq analz(T_1) \bigcup analz(T_2)$. Hence, we get $analz(T_1) \bigcup analz(T_2) = analz(T_1 \bigcup T_2)$.

4. Suppose $k \in synth(T)$. We have by definition $T \subseteq synth(T)$. Now let Y be the set $synth(T)\backslash\{k\}$, *i.e.*, the set obtained by removing k from $synth(T)$. Now if $k \notin T$, then we get $T \subseteq Y$.

   Also if $u \in Y$ and $v \in Y$, then $u, v \in synth(T)$. By definition, $\langle u, v\rangle \in synth(T)$ and since key k is not a pair, we get $< u, v > \in Y$. Similarly, we can show $PCS(k', u, k_1, k_t), FakeSign(k', u, k_1, k_t) \in Y$ if $u, k'^{-1} \in Y$, where k'^{-1} is the signature key corresponding to the verification key k', and $sig(k', u), tsig(k', k_1, u) \in Y$ if $u, k'^{-1}, PCS(k', u, k_1, k_t) \in Y$ where k'^{-1} is the signature key corresponding to the verification key k'.

   Since $T \subseteq Y$, we get by definition, $synth(T) \subseteq Y$. A contradiction since Y is the set $synth(T)\backslash\{k\}$. Therefore $k \in T$.

5. Suppose $PCS(k_1, m, k_2, k_3) \in synth(analz(T))$. We will show that if $k_1^{-1} \notin analz(T)$, then $PCS(k_1, m, k_2, k_3) \in analz(T)$.

   We have by definition $analz(T) \subseteq synth(analz(T))$. Now, let Y be the set $synth(analz(T))\backslash\{PCS(k_1, m, k_2, k_3)\}$, *i.e.*, the set obtained by removing $PCS(k_1, m, k_2, k_3)$ from $synth(analz(T))$. We have by definition, $analz(T) \subseteq synth(analz(T))$. Now if $PCS(k_1, m, k_2, k_3) \notin analz(T)$, then $analz(T) \subseteq Y$. Also since ${k_1}^{-1} \notin analz(T)$, by part 4, ${k_1}^{-1} \notin synth(analz(T))$.

   If $u, k'^{-1} \in Y$, then $u, k'^{-}1 \in synth(analz(T))$. By definition of $synth$, $PCS(k', u, k", k_t) \in synth(analz(T))$ for any public keys $k", k_t$. Since ${k_1}^{-1} \notin$

$synth(analz(T))$, k_1 and k' are different. Therefore, $PCS(k', u, k", k_t)$ and $PCS(k_1, m, k_2, k_3)$ are different and we get $PCS(k', u, k", k_t) \in Y$.

Also if $u \in Y$ and $v \in Y$, then $u, v \in synth(analz(T))$. By definition, $< u, v >\in synth(analz(T))$ and since $PCS(k_1, m, k_2, k_3)$ is not a pair, we get $< u, v >\in Y$. Similarly, we can show $FakeSign(k', u, k", k_t) \in Y$ if $u, k'^{-1} \in Y$, where k'^{-1} is the signature key corresponding to the verification key k', and

$sig(k', u), tsig(k', k_1, u) \in Y$ if $u, k'^{-1}, PCS(k', u, k_1, k_t) \in Y$ where k'^{-1} is the signature key corresponding to the verification key k'.

Hence by definition, $synth(analz(T)) \subseteq Y$. This is a contradiction since Y is the set $synth(analz(T))\backslash\{PCS(k_1, m, k_2, k_3)\}$. Therefore, we must have $PCS(k_1, m, k_2, k_3) \in analz(T)$.

6. Proof is similar to part 5.

7. Proof is similar to part 5.

$\square$

Proposition 3.4.2. *For any set of terms T,*

1. *If a pair $\langle u, v\rangle \in synth(T)$, then either $\langle u, v\rangle \in synth(T)$ or $u, v \in synth(T)$.*

2. *If $PCS(k_1, m, k_2, k_3) \in synth(T)$ then either $PCS(k_1, m, k_2, k_3) \in T$ or $u \in synth(T)$.*

3. If $FakeSign(k, u, k_1, k_t) \in synth(T)$ then either $FakeSign(k, u, k_1, k_t) \in T$ or $u \in synth(T)$.

4. If $sig(k, m) \in synth(T)$ then either $sig(k, m) \in T$ or $m \in synth(T)$.

5. If $tsig(k, k_1, m) \in synth(T)$ then either $tsig(k, k_1, m) \in T$ or $m \in synth(T)$.

Proof. We shall show part 1. The proofs of parts 2, 3, 4 and 5 are similar to the proof of part 1.

Suppose $\langle u, v\rangle \in synth(T)$. We have by definition $T \subseteq synth(T)$. Now let Y be the set $synth(T)\backslash\{\langle u, v\rangle\}$, *i.e.*, the set obtained by removing $\langle u, v\rangle$ from $synth(T)$. If $\langle u, v\rangle \notin T$ then $T \subseteq Y$. We have to show that $u, v \in synth(T)$ in that case.

Suppose $u \notin synth(T)$. Now if $u' \in Y$ and $v' \in Y$, then $u', v' \in synth(T)$ ($Y \subset synth(T)$). Since $u \notin synth(T)$, we get u and u' are different and hence the pairs $\langle u, v\rangle$ and $\langle u', v'\rangle$ are different. By definition of *synth*, we get $\langle u', v'\rangle \in synth(T)$. Hence $\langle u', v'\rangle \in Y$ (Since $Y = synth(T)\backslash\{\langle u, v\rangle$).

Similarly, it can be shown that $PCS(k', u', k_1, k_t)$, $FakeSign(k', u', k_1, k_t) \in Y$ if $u', k'^{-1} \in Y$, where k'^{-1} is the signature key corresponding to the verification key k', and

$sig(k', u')$, $tsig(k', k_1, u') \in Y$ if $u, k'^{-1}, PCS(k', u', k_1, k_t) \in Y$ where k'^{-1} is the signature key corresponding to the verification key k'. Hence by definition, $synth(T) \subseteq Y$. A contradiction since Y is the set $synth(T)\backslash\{\langle u, v\rangle\}$. Therefore $u \in synth(T)$. Similarly, we can show $v \in synth(T)$. $\square$

Proposition 3.4.3. *For any set of terms T,*

$analz(synth(analz(T))) = synth(analz(T)))$ *and* $synth(analz(synth(analz(T))))$ $= synth(analz(T)))$.

Proof. By definition, $synth(analz(T)) \subseteq analz(synth(analz(T)))$. Clearly, we have $synth(analz(T)) \subseteq synth(analz(T))$.

Now, if $\langle u, v\rangle \in synth(analz(T)))$, then by part 1 we get <u>either</u> $\langle u, v\rangle \in analz(T)$ <u>or</u> $u, v \in synth(analz(T))$. If $\langle u, v\rangle \in analz(T)$, then by definition of $analz$, $u, v \in analz(T)$. Since $analz(T) \subseteq synth(analz(T)$, we get $u, v \in synth(analz(T))$. Therefore if $\langle u, v\rangle \in synth(analz(T)))$, then $u, v \in synth(analz(T))$.

Similarly, $u \in synth(analz(T))$ if $sig(k, u) \in synth(analz(T))$,
$u \in synth(analz(T))$ if $PCS(k, u, k_1, k_t) \in synth(analz(T))$,
$u \in synth(analz(T))$ if $FakeSign(k, u, k_1, k_t) \in synth(analz(T))$, and
$u \in synth(analz(T))$ if $tsig(k, k_1, u) \in synth(analz(T))$.

By definition of $analz$, we get $analz(synth(analz(T))) \subseteq synth(analz(T))$. Therefore, $analz(synth(analz(T))) = synth(analz(T))$.

Now, by part 1 of proposition 3.4.1 part 1 of proposition 3.4.2, we get $synth(analz(synth(analz(T)))) \subseteq synth(analz(T))$.

Hence, we get $synth(analz(synth(analz(T)))) = synth(analz(T))$. □

Now, we are ready to prove proposition 3.1.2.

Proposition 3.1.2. *If S, S' are states such that S' is obtained from S by the application of rule in* **I** *other than GEN, then $W(S) = W(S')$.*

Proof. We shall prove this by inspection of the rules used. Recall that for a given state S_1, the set of messages available to the intruder is defined to be $A(S_1) = \{u|C(u), D(u), M(u), N(u), Rn(k_1, k_2, u) \in S_1\}$. The view of the intruder is defined to be $W(S) = synth(analz(A(S_1)))$.

First of all if the transition from S to S' involves an I/O rule, *i.e.*, one of the rules $REC, SND, REC_R, SND_R, SND_T$, then the set of available messages in S and S' are the same. Hence the view of the intruder is the same in S and S'.

Now suppose the transition from S to S' uses decomposition rule $DCMP$, in which a pair $\langle u, v\rangle$ in the decomposition memory of the intruder is split. Then the set of available messages increases from S to S'. In particular, we have $A(S') = A(S) \cup \{u, v\}$. We also have $\langle u, v\rangle \in A(S)$. By part 3 of proposition 3.4.1, $analz(A(S')) = analz(A(S)) \cup analz(u) \cup analz(v)$. By definition, since $\langle u, v\rangle \in A(S)$, we get $u, v \in analz(A(S))$. By part 1 of proposition 3.4.1, we get $analz(u) \subseteq analz(A(S))$ and $analz(v) \subseteq analz(A(S))$. Hence $analz(A(S')) = analz(A(S))$. Therefore, the view of the intruder in S' is the same as in S'.

Similarly, if the rules $ReadPCS$, $ReadSig$, $ReadFakeSig$, $ReadTPSig$ are used, the view of the intruder in S' is the same as in S'. If the decomposition rule LRN is used, the set of available messages do not change and once again we get $W(S) = W(S')$.

Now suppose the transition from S to S' uses composition rule CMP, in which a pair $\langle u, v\rangle$ is formed using terms u, v in the decomposition memory of the intruder.

Then the set of available messages increases from S to S', and hence $W(S) \subseteq W(S')$. In particular, we have $A(S') = A(S) \cup < u, v >$. We also have $u, v \in A(S)$. By part 3 of proposition 3.4.1, $analz(A(S')) = analz(A(S)) \cup analz(\{\langle u, v\rangle\})$. Clearly by definition, we have $analz(A(S)) \subseteq W(S)$. Also since $u, v \in A(S)$, we get $u, v \in analz(A(S))$, and by definition of *synth*, $\langle u, v\rangle \in W(S)$. Hence we get by part 1 of proposition 3.4.1 $analz(\{\langle u, v\rangle\}) \subseteq analz(W(S))$. By proposition 3.4.3, $analz(W(S)) = W(S)$ and therefore we get $analz(\{\langle u, v\rangle\}) \subseteq W(S)$. Hence, $analz(A(S')) \subseteq W(S)$. By part 1 of proposition 3.4.1, we obtain $W(S') \subseteq W(S)$. Therefore, $W(S) = W(S')$.

Similarly, if the composition rules PCS, $FakeSign$, SIG and $TPSIG$ are used, the view of the intruder in S' is the same as in S'. If the composition rule USE is used, the set of available messages do not change and once again we get $W(S) = W(S')$. □

Now, we prove proposition 3.1.3.

Proposition 3.1.3. *Suppose* Σ *contains* $HonestGuy(k, k^{-1})$ *or* $TTP(k, k^{-1})$ *for some* k, k^{-1}*. Then for all reachable states* S*,* k *is not in the view of the intruder.*

Proof. Intuitively, protocol messages do not require an honest signer and trusted third party to send any private keys. Hence, their private keys are never released.

We shall prove the prove the proposition for the case $TTP(k, k^{-1}) \in \Sigma$. The case when $HonestGuy(k, k^{-1}) \in \Sigma$ can be similarly proved. In view of proposition 3.4.1, if k is in the view of the intruder at S, *i.e.*, if it is in $synth(analz(A(S))$, then k

must be in $analz(A(S))$. We show by induction on the number of steps that it takes to reach S from Σ that $k \notin analz(A(S))$.

Base case: S is Σ. Now, the only available messages in Σ are private keys of dishonest signers and preagreed-texts. Clearly, since $k \notin A(\Sigma)$, and $analz(A(\Sigma)) = A(\Sigma)$. Hence $k \notin analz(A(\Sigma))$.

Induction hypothesis: Suppose it is the case that for all states S reachable from Σ in $\leq l$ steps, $k \notin analz(A(S))$.

Consider a state S' reachable from Σ in $l + 1$ steps. We must have there is a state S such that

a) S' is obtained from S by a single transition.

b) S is obtained from Σ in $\leq l$ steps.

By induction hypothesis, it must be case that $k \notin analz(A(S))$ Now we show by cases on the transition rule being applied to obtain S' from S that $k \notin analz(A(S'))$.

In case the role generation rule RG is used then the set of available messages increases by a new constant n' of the sort $unique\_identifier$. Clearly k is not the same as n' and $analz(\{n'\}) = \{n'\}$. Hence $k \notin analz(\{n'\})$. Now $A(S') = A(S) \cup \{n'\}$.

By proposition 3.4.1, we get $analz(A(S')) = analz(A(S)) \cup analz(\{n'\})$. Since $k \notin analz(A(S))$, we get $k \notin analz(A(S'))$.

If the rule used is a transition in $\mathbf{I}$ other than GEN, $k \notin analz(A(S'))$ if $k \notin analz(A(S))$. If GEN is used then the set of available messages increases by a new constant u of the sort $mssg$. Clearly k is not the same as u and $analz(\{u\}) = \{u\}$. Hence $k \notin analz(\{u\})$. Now $A(S') = A(S) \cup \{u\}$. By proposition 3.4.1, we get $analz(A(S')) = analz(A(S)) \cup analz(\{u\}$. Since $k \notin analz(A(S))$, we get $k \notin analz(A(S'))$.

If the transition rule O_1 is used, then the set of available messages increases by a term $PCS(k_1, pd, k_2, k_3)$, where k_1, k_2, k_3 are terms of the sort *public_key*, and *pd* is $\langle m', n', k_1, k_2, k_3\rangle$ for some terms m', n' of the sort *preagreed_text* and $unique\_identifier$ respectively. It can be shown that $analz(PCS(k_1, pd, k_2, k_3))= PCS(k_1, pd, k_2, k_3) \cup analz\{pd\}$. Clearly, k is not the same as $PCS(k_1, pd, k_2, k_3)$. It can be shown that $analz(pd)$ is the set that consists of $< m', n', k_1, k_2, k_3 >$, $< m', n', k_1, k_2 >$, $< m', n', k_1 >$, $< m', n' >$, m', n', k_1, k_3. None of these are the terms of the sort *private_key*, and we get $k \notin analz(pd)$. Therefore, $k \notin analz(\{PCS(k_1, pd, k_2, k_3)\})$. We also have $A(S') = A(S) \cup \{PCS(k_1, pd, k_2, k_3)\}$. By proposition 3.4.1, we get $analz(A(S')) = analz(A(S)) \cup analz(\{PCS(k_1, pd, k_2, k_3)\})$. Since $k \notin analz(A(S))$, we get $k \notin analz(A(S'))$.

Similarly, if any other rule in $\mathbf{O} \cup \mathbf{R} \cup \mathbf{T}$ is used, we get $k \notin analz(A(S'))$ if $k \notin analz(A(S))$.

Hence, by induction $k \notin analz(A(S))$ for all states S reachable from Σ. $\square$

3.4.2 Database properties

We are ready to show proposition 3.2.1.

Proposition 3.2.1. *If S is a state reachable from S_1, then S contains exactly one of the following:*

$T_0(pd)$, $T_{ab}(pd, aborted, ab\_tok)$, $T_{or}(pd, resolved, res\_cn)$, $T_{rr}(pd, resolved, res\_cn)$, $T_{abf}(pd, aborted, ab\_tok)$, $T_{orf}(pd, resolved, res\_cn)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$.

Furthermore, there is exactly one occurrence of the fact $T_i(pd, -)$ in S for each $i \in \{0, ab, or, abf, orf, rrf1, rrf2\}$.

Proof. Intuitively, the freshness of the identifier n guarantees that there is only one instance of a fact of the form $T_i(pd, -)$ for $i \in \{0, or, ab, abf, orf, rrf1, rrf2\}$. Furthermore, the rules in the protocol theory $\mathbf{T}$ ensure that whenever one fact $T_i(pd, -)$ is consumed, another fact $T_j(pd, -)$ is created.

We prove the proposition by induction on the number of transitions it takes to reach S from S_1.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. Since the identifier n is freshly generated in this transition, S_0 does not contain any fact of the form $T_i(pd, -)$ for $i \in \{0, or, ab, abf, orf, rrf1, rrf2\}$. In the transition $T_0(pd)$ is created. No other fact of the form $T_i(pd, -)$ is created. Hence, the statement of the proposition is true.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l + 1$ steps. Then there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S by the application of t using σ.

2) S' is reachable from S_1 by l steps.

By Induction hypothesis, S contains exactly one fact of the following:

$T_0(pd)$, $T_{ab}(pd, aborted, ab\_tok)$, $T_{or}(pd, resolved, res\_cn)$, $T_{rr}(pd, resolved, res\_cn)$, $T_{abf}(pd, aborted, ab\_tok)$, $T_{orf}(pd, resolved, res\_cn)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$.

Furthermore, there is exactly one occurence of the fact $T_i(pd, -)$ in S for each $i \in \{0, ab, or, abf, orf, rrf1, rrf2\}$.

We shall show that if S contains $T_0(pd)$, then the statement of the proposition holds in S' also. The other cases can be treated similarly.

Now, if the transition from S to S' uses a transition in $\mathbf{O} \cup \mathbf{R} \cup \mathbf{I}$, then since no T_i fact is created or consumed in this transition, the statement of the proposition still holds in S'.

If the transition from S to S' uses the transition RG, then a $T_0(pd')$ fact is created, where $pd' =< k_1, m_1, n', k_2, k_t >$. n' is freshly generated and is different from the identifier n in pd. Hence, pd and pd' are different. No other T_i fact is created or consumed. Hence, the statement of the proposition is true in S'.

If the transition from S to S' uses the transition T_{ab}, then a $T_0(pd')$ fact is consumed and a fact of the form $T_{ab}(pd', -)$ is created. No other T_i fact is created or consumed. Now, there are two cases: pd' is the same as pd, or pd is different from

pd'. If pd' is the same as pd, then $T_0(pd)$ is consumed and $T_{ab}(pd, aborted, ab\_tok)$ is created, and the statement of the proposition is true. If pd is different from pd', then no fact of the form $T_i(pd, -)$ is created or consumed and the proposition is still true. Similarly, if the transition from S to S' uses the transition T_{or} or T_{rr}, then the statement of the proposition is true in S'.

If the transition from S to S' uses the transition T_{abf}, then a fact of the form $T_{ab}(pd', -)$ is consumed and a fact of the form $T_{abf}(pd', -)$ is created. No other fact of the sort T_i is created or consumed. Since S contains $T_0(pd)$, by induction hypothesis, S does not contain $T_{ab}(pd, -)$. Hence pd must be different from pd', and the statement of the proposition is true in S'. Similarly, if the transition from S to S' uses one of the transitions $T_{orf}, T_{rrf1}, T_{rrf2}$, then the statement of the proposition is true in S'.

Hence, if S contains $T_0(pd, -)$, the statement of the proposition is true in S'.

By induction, the proposition is true. $\square$

Now we are ready to prove properties of database persistence and consistency:

Lemma 3.2.3 1. *For all states S reachable from S_1, one of the following is true:*

1) T has no entry for pd in S.

2) T has an abort for pd in S.

3) T has a resolution for pd in S.

If T has an abort for pd, then for all states S' reachable from S, T has an abort for pd in S'. If T has a resolution for pd, then for all states S' reachable from S, T

has a resolution for pd *in* S'.

Proof. By proposition 3.2.1, it follows immediately that one of the following is true:

1) T has no entry for pd in S.

2) T has an abort for pd in S.

3) T has a resolution for pd in S.

Now, suppose that T has an abort for pd in S. We need to show that for all states S' reachable from S, T has an abort for pd. We show this induction on the number of steps it takes to reach S' from S.

Base case: S' is S. Clearly by assumption T has an abort for pd is S'.

Induction Hypothesis: Let T have an abort for pd for all states reachable from S by l steps. Let S" be a state reachable from S by $l + 1$ steps. Then there is a state S', a transition rule t, and a ground substitution σ such that

1) S" is obtained from S' by the application of t using σ.

2) S" is reachable from S by l steps.

3) By induction hypothesis, T has an abort for pd in S'.

Hence S' contains $T_{ab}(pd, aborted, ab\_tok)$ or $T_{abf}(pd, aborted, ab\_tok)$.

An inspection of rules tells that a T_{abf} fact is never consumed, hence if S' contains $T_{abf}(pd, aborted, ab\_tok)$, then S" also contains $T_{abf}(pd, aborted, ab\_tok)$. Also if $T_{ab}(pd, aborted, ab\_tok)$ is consumed in the transition from S' to S", then t must be T_{abf}, and S" must contain $T_{abf}(pd, aborted, ab\_tok)$. Therefore S" must contain $T_{ab}(pd, aborted, ab\_tok)$ or $T_{abf}(pd, aborted, ab\_tok)$ if

$T_{ab}(pd, aborted, ab\_tok) \in S'$.

Hence in S", T contains an abort for pd. By induction T contains an abort for pd for all reachable for S.

Similarly, if T has a resolution for pd, then for all states reachable from S, T has a resolution for pd. □

Lemma 3.2.4. ***Database Consistency***: *For all states S reachable from S_1, the following are true:*

1) If T has no entry for pd in S, then T does not have an abort or a resolution for pd in S.

2) If T has an abort for pd in S, then T does not have a resolution for pd in S.

3) If T has a resolution for pd in S, then T does not have an abort for pd in S.

Proof. The lemma follows immediately from proposition 3.2.1 since every state S reachable from S_1 contains exactly one of the following:

$T_0(pd)$, $T_{ab}(pd, aborted, ab\_tok)$, $T_{or}(pd, resolved, res\_cn)$, $T_{rr}(pd, resolved, res\_cn)$, $T_{abf}(pd, aborted, ab\_tok)$, $T_{orf}(pd, resolved, res\_cn)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$. □

Now, we shall prove lemma 3.2.5. We need the following propositions.

Proposition 3.4.4. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in $\mathbf{O} \cup \mathbf{R}$. If an abort_token or a resolution for pd is not in the view of the intruder at S, then an abort token or a resolution for pd is not in the view of the intruder at S' also.*

Proof. Intuitively, since the intruder does not learn the private key of T, the only way intruder can get hold of an abort_token or a resolution for pd is when T sends them.

By proposition 3.1.3, the private key k_t is not in the view of the intruder at S, S'. Hence if $A(S)$ and $A(S')$ are the sets of available messages at S and S'respectively, by proposition 3.4.1, an abort_token or a resolution for pd is in the view of the intruder at S, S' only if they are elements of $analz(A(S)), analz(A(S'))$ respectively.

If they are not in the view of the intruder at S, then they are not in $analz(A(S))$. We shall show by cases that if the rule applied to obtain S' from S is in $\mathbf{R}$, then that neither of these can be in $analz(A(S'))$. The case when the rule applied is in $\mathbf{R}$ can be treated similarly.

1. Rule O_1: In this case, $PCS(k_1, pd', k_2, k_3)$, is deposited on the network where k_1, k_2, k_3 are keys of the sort *public_key* and $pd' =< m', n', k_1, k_2, k_3 >$. m' and n' are terms of the sort *preagreed_text* and *unique_identifier* respectively. (Note pd' may not be the same as pd: there may be several concurrent executions of the protocol). By proposition 3.4.1, $analz(A(S')) = analz(A(S)) \cup analz(\{PCS(k_1, pd', k_2, k_3)\})$. Now, $analz(\{PCS(k_1, pd', k_2, k_3)\}) = \{PCS(k_1, pd', k_2, k_3)\} \cup analz(pd')$. $analz(S)$ does not contain an abort_token or resolution for pd. Clearly, $analz(\{PCS(k_1, pd', k_2, k_3)\})$ also does not contain either of these and hence an abort_token or a resolution is not in $analz(A(S'))$.

2. Rule O_3: A message of $sig(k_1, < m', n', k_1, k_2, k_3 >)$ is deposited on the net-

work. $analz(sig(k_1, < m', n', k_1, k_2, k_3 >))$ does not contain an abort_token (an abort_token is a signature on an abort request). It also does not contain a resolution (a resolution is a pair of $tsig$ terms). Once again an abort_token or a resolution for pd is not in $analz(A(S'))$.

3. Rule $O_{ab?}$: A message $sig(k_1, < abort, < m', n', k_1, k_2, k_3 >>)$ is deposited on the channel between k_1 and k_3. As in case 2, once again an abort_token or a resolution for pd is not in $analz(A(S'))$.

4. Rule $O_{res?}$: Similar to previous cases.

5. Rule O_2, O_{com}, O_{ab1}, O_{ab2}, O_{res1}, O_{res2} : The set of available messages actually decreases, and hence an abort_token or a resolution for pd is not in $analz(A(S'))$.

□

Proposition 3.4.5. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* **I**. *If an abort_token or a resolution for pd is not in the view of the intruder at S, then an abort token or a resolution for pd is not in the view of the intruder at S' also.*

Proof. Once again, the private key of T is not in the view of the intruder at S, S'. By proposition 3.4.1 an abort_token or a resolution for pd is in the view of the intruder at S, S' only if they are in $analz(A(S))$, $analz(A(S'))$ respectively.

By proposition 3.1.2, view of the intruder does not change by an application of a rule in **I** other than GEN. Now, if GEN is used, then the set of available messages increases by a new constant u of the sort $mssg$. Since $analz(u) = \{u\}$, we get abort token or a resolution is in the view of the intruder at S' only if it is in the view of the intruder at S. $\square$

Proposition 3.4.6. *Let S, S' be reachable states such that S' is obtained from S by the application of the role generation rule RG. If an abort_token or a resolution for pd is not in the view of the intruder at S, then an abort token or a resolution for pd is not in the view of the intruder at S' also.*

Proof. Once again, since the private key of T is not in the view of the intruder at S, S'. By proposition 3.4.1 an abort_token or a resolution for *pd* is in the view of the intruder at S, S' only if they are in $analz(A(S))$, $analz(A(S'))$ respectively.

If GEN is used, then the set of available messages increases by a new constant n' of the sort $unique\_identifier$. Since $analz(n') = \{n'\}$, we get an abort_token or a resolution is in the view of the intruder at S' only if it is in the view of the intruder at S. $\square$

Proposition 3.4.7. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* $\mathbf{T}$. *If an abort_token or a resolution for pd is not in the view of the intruder at S, then*

1) An abort_token for pd is in the view of the intruder at S' only if T has an abort_token for pd in S'.

2) A resolution for pd is in the view of the intruder at S' only if T has a resolution for pd in S'.

Proof. Intuitively, since the intruder does not learn the private key of T, the only way intruder can get hold of an abort_token or a resolution for *pd* is when T sends them. T sends an abort_token or a resolution only if it already has it in its database or if it creates the abort_token or a resolution in response to a request. T always stores whatever decisions it takes in response to a request.

The proof is again by cases on the rule being used in the transition from S to S'. We consider one such case where the rule T_{ab} is used. The other cases can be similarly dealt with. As the key k_t is not in the view of the intruder at S, S' (proposition 3.1.3), an abort_token or resolution is in the view of the intruder only if they are contained in $analz(A(S)), analz(A(S'))$ respectively.

If the rule T_{ab} is used, then the only new available message at S' is $sig(k_3, sig(k_1, \langle abort, pd'\rangle))$ where $pd' =< m', n', k_1, k_2, k_3 >$. k_1, k_2, k_3 are terms of sort $public\_key$, m' is a term of the sort $preagreed\_text$ and n' is a term of the sort $unique\_identifier$. An abort entry for pd' is thus created, *i.e.*, $T_{ab}(pd', sig(k_3, sig(k_1, \langle abort, pd'\rangle)))$ is contained in S'.

Now it can be easily shown that $analz(sig(k_3, sig(k_1, \langle abort, pd'\rangle)))$ is the set that consists of: $sig(k_3, sig(k_1, \langle abort, pd'\rangle)), sig(k_1, \langle abort, pd'\rangle), \langle abort, pd'\rangle, abort, pd'$, k_1, k_2, m', n' and k_3. Clearly, the abort_token for *pd* is in $analz(A(S'))$ only if it is in $analz(A(S))$ or pd' is the same as *pd*. A resolution for *pd* is in the view of the intruder only if it is $analz(A(S))$. By hypothesis, none of these are in $analz(A(S))$.

Since the abort_token is not in the view of the intruder at S, abort_token for *pd* is in $analz(A(S'))$ only if *pd'* is the same as *pd*. In that case, T has an abort_token for *pd*.

□

Now, we are ready to prove 3.2.5.

Lemma 3.2.5. *For all states S reachable from S_1, an abort_token for pd is in the view of the intruder only if T has an abort_token for pd. A resolution for pd is in the view of the intruder only if T has a resolution for pd.*

Proof. Intuitively, by propositions 3.4.4, 3.4.5, 3.4.6, 3.4.7 if a state transition produces an abort_token or resolution that was not present before, then T must have the corresponding decision. Furthermore, once T creates an entry for *pd*, then by database persistence 3.2.3, it maintains that entry forever.

We prove this by induction on number of steps that it takes to go from S_1 to S.

Base case: S is S_1. Since the identifier n is freshly generated in the transition from S_0 to S_1, clearly an abort_token or a resolution for *pd* is not in the view of the intruder at S_0. Hence by proposition 3.4.6, an abort_token or a resolution for *pd* is not in the view of the intruder at S_1 also.

Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps. Consider a state S' such that S' is reachable from S_1 in $l + 1$ steps and an abort_token or a resolution is in the view of the intruder.

Since S' is reachable from S_1 in $l+1$ steps, there must be a state S such that:

1) S' is reachable from S by a single transition, and

2) S is reachable from S_1 by l steps.

Now the following 3 cases are possible:

1) An abort_token for *pd* is in the view of the intruder at S. Then by induction hypothesis, T contains an abort entry for *pd* at S. By database persistence(proposition 3.2.3), T also contains an abort entry for *pd* at S'.

2) A resolution for *pd* is in the view of the intruder at S. Then again by induction hypothesis and database persistence(proposition 3.2.3), T also contains a resolution for *pd* at S'.

3) Neither an abort_token or a resolution for *pd* is in the view of the intruder at S. Then if S' contains an abort_token for *pd*, then by propositions 3.4.4, 3.4.5, 3.4.6 the transition from S to S' must use a rule in **T**. In this case, by proposition 3.4.7, T contains an abort_token for *pd* in S'. Similarly, if a resolution for *pd* is in the view of the intruder at S', T contains a resolution for *pd*. □

3.4.3 Effectiveness for honest O

We prove the lemmas needed to prove effectiveness of honest O. We start by proving proposition 3.2.6.

Lemma 3.2.6. *If S is a state reachable from S_1, then S contains exactly one of the following:*

$O_0(pd)$, $O_1(pd, me_1)$, $O_2(pd, me_1, me_2)$, $O_3(pd, me_1, me_2, me_3)$,

$O_{com}(pd, me_1, me_2, me_3, me_4)$, $O_{ab?}(pd, me_1, ab\_req)$,

$O_{res?}(pd, me_1, me_2, me_3, res\_req)$, $O_{ab1}(pd, me_1, ab\_req, ab\_tok)$,

$O_{res1}(pd, me_1, ab\_req, res\_cn)$, $O_{ab2}(pd, me_1, me_2, me_3, ab\_req, ab\_tok)$,

$O_{res2}(pd, me_1, me_2, me_3, res\_req, res\_cn)$.

Furthermore, there is exactly one occurrence of the fact $O_i(pd, -)$ for each $i \in \{0, 1, 2, 3, ab?, ab1, ab2, res?, res1, res2, com\}$.

Proof. Intuitively, the freshness of the globally unique identifier guarantees that there is only one instance of a fact of the form $O_i(pd, -)$. Furthermore, the rules in the protocol theory **O** ensure that whenever one fact $O_i(pd, -)$ is consumed, another fact $O_j(pd, -)$ is created.

We prove the proposition by induction on the length of reachability of S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. Since the identifier n is freshly generated in this transition, S_0 does not contain any fact of the form $O_i(pd, -)$ for $i \in \{0, 1, 2, 3, ab?, ab1, ab2, res?, res1, res2, com\}$. In the transition $O_0(pd)$ is created. No other fact of the form $O_i(pd, -)$ is created. Hence, the statement of the proposition is true.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l + 1$ steps. Then, there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S is by the application of t using σ.

2) S is reachable from S_1 by l steps.

By Induction hypothesis, S contains exactly one fact of the following:

$O_0(pd)$, $O_1(pd, me_1)$, $O_2(pd, me_1, me_2)$, $O_3(pd, me_1, me_2, me_3)$,

$O_{com}(pd, me_1, me_2, me_3, me_4)$, $O_{ab?}(pd, me_1, ab\_req)$,

$O_{res?}(pd, me_1, me_2, me_3, res\_req)$, $O_{ab1}(pd, me_1, ab\_req, ab\_tok)$,

$O_{res1}(pd, me_1, ab\_req, res\_cn)$, $O_{ab2}(pd, me_1, me_2, me_3, ab\_req, ab\_tok)$,

$O_{res2}(pd, me_1, me_2, me_3, res\_req, res\_cn)$.

Furthermore, there is exactly one occurence of the fact $O_i(pd, -)$ for each

$i \in \{0, 1, 2, 3, ab?, ab1, ab2, res?, res1, res2, com\}$.

We shall show that if S contains $O_0(pd)$, then the statement of the proposition holds in S' also. The other cases can be treated similarly. Now, if the transition from S to S' uses a transition in $\mathbf{T} \cup \mathbf{R} \cup \mathbf{I}$, then since no O_i fact is created or consumed in this transition, the statement of the proposition still holds in S'.

If the transition from S to S' uses the transition RG, then a fact $O_0(pd')$ is created, where $pd' =< k_1, m_1, n', k_2, k_t >$, where n' is freshly generated and is different from the globally unique identifier in pd. Hence, pd and pd' are different. No other O_i fact is created or consumed. Hence, the statement of the proposition is true in S'.

If the transition from S to S' uses the transition O_1, then a fact $O_0(pd')$ is consumed and a fact of the form $O_1(pd', -)$ is created. No other O_i fact is created or consumed. Now, there are two cases: pd' is the same as pd, or pd is different from pd'. If pd is the same as pd, then $O_0(pd)$ is consumed and $O_1(pd, me_1)$ is created, and the statement of the proposition is true. If pd is different from pd', then no fact of the form $O_i(pd, -)$ is created or consumed in the transition and the proposition is still true.

If the transition from S to S' uses the transition $O_{ab?}$, then then a fact $O_1(pd', -)$ is consumed and a fact of the form $O_{ab?}(pd', -)$ is created. No other fact O_i is created or consumed. Since S contains $O_0(pd)$, S' contains $O_0(pd)$. By induction hypothesis, there is only one occurence of the form $O_i(pd, -)$. Hence S the only fact of the form contained by S is $O_0(pd)$.

In particular, S cannot contain $O_1(pd, -)$ and hence the transition to S' does not produce a fact of the form $R_i(pd, -)$. Hence the statement of the proposition is true in S'. Therefore, pd must be different from pd'. and the statement of the proposition is true in S. Similarly, if the transition from S' to S uses one of the transitions: O_2, O_3, O_{com}, $O_{ab?}$, $O_{res?}$, O_{ab1}, O_{ab2}, O_{res1}, or O_{res2}, then the statement of the proposition is true in S'. Therefore if S contains $O_0(pd, -)$, the statement of the proposition is true in S'.

Hence, by induction, the proposition is true. $\square$

Proposition 3.4.8. *Let S and S' be states reachable from S_1 such that S' is obtained from S by an application of a single rule. Then*

1. If $O_0(pd) \in S$, then $O_0(pd) \in S'$ or $O_1(pd, -) \in S'$. If $O_0(pd) \in S'$, then

$O_0(pd) \in S$.

2. *If $O_1(pd, -) \in S$, then $O_1(pd, -) \in S'$ or $O_{ab?}(pd, -) \in S'$ or $O_2(pd, -) \in S'$. If $O_1(pd, -) \in S'$, then $O_0(pd) \in S$ or $O_1(pd, -) \in S$.*

3. *If $O_2(pd, -) \in S$, then $O_2(pd, -) \in S'$ or $O_3(pd, -) \in S'$. If $O_2(pd, -) \in S'$, then $O_1(pd, -) \in S$ or $O_2(pd, -) \in S$.*

4. *If $O_3(pd, -) \in S$, then $O_3(pd, -) \in S'$ or $O_{res?}(pd, -) \in S'$ or $O_{com}(pd, -) \in S'$. If $O_3(pd, -) \in S'$, then $O_2(pd) \in S$ or $O_3(pd, -) \in S$.*

5. *If $O_{com}(pd, -) \in S$, then $O_{com}(pd, -) \in S'$. If $O_{com}(pd, -) \in S'$, then $O_3(pd, -) \in S$ or $O_{com}(pd, -) \in S$.*

6. *If $O_{ab?}(pd, -) \in S$, then $O_{ab?}(pd, -) \in S'$ or $O_{ab1}(pd) \in S'$ or $O_{res1}(pd, -) \in S'$. If $O_{ab?}(pd, -) \in S'$ then $O_1(pd, -) \in S$ or $O_{ab?}(pd, -) \in S$.*

7. *If $O_{res?}(pd, -) \in S$, then $O_{res?}(pd, -) \in S'$ or $O_{ab2}(pd) \in S'$ or $O_{res2}(pd, -) \in S'$. If $O_{res?}(pd, -) \in S'$ then $O_3(pd, -) \in S$ or $O_{res?}(pd, -) \in S$.*

8. *If $O_{ab1}(pd, -) \in S$, then $O_{ab1}(pd, -) \in S'$. If $O_{ab1}(pd, -) \in S'$ then $O_{ab?}(pd, -) \in S$ or $O_{ab1}(pd, -) \in S$.*

9. *If $O_{res1}(pd, -) \in S$, then $O_{res1}(pd, -) \in S'$. If $O_{res1}(pd, -) \in S'$ then $O_{ab?}(pd, -) \in S$ or $O_{res}(pd, -) \in S$.*

10. *If $O_{ab2}(pd, -) \in S$, then $O_{ab2}(pd, -) \in S'$. If $O_{ab2}(pd, -) \in S'$ then $O_{res?}(pd, -) \in S$ or $O_{ab2}(pd, -) \in S$.*

11. *If $O_{res2}(pd,-) \in S$, then $O_{res2}(pd,-) \in S'$. If $O_{res2}(pd,-) \in S'$ then $O_{res?}(pd, -) \in S$ or $O_{res2}(pd,-) \in S$.*

Proof. We shall show part 1. The other parts can be similarly proved. If $O_0(pd) \in S$, then an inspection of rules tells that the only way $O_0(pd)$ can be consumed in the transition to S' is if the rule O_1 is used. Now, if O_1 is used and $O_0(pd,-)$ consumed in the transition, then an inspection of the rule tells that a fact $O_1(pd,-)$ is created. Hence, if $O_0(pd) \in S$, then $O_0(pd) \in S'$ or $O_1(pd,-) \in S'$.

Now suppose $O_0(pd) \in S'$ and $O_0(pd) \notin S$. Then, since an O_0 fact is created in the transition to S', an inspection of the rules tells that RG must be used in the transition from S to S'. In this transition, $O_0(pd')$ is created. Now, in the transition the identifier in pd' is freshly generated and must be different from pd. Hence $O_0(pd')$ is different from $O_0(pd)$ and we get $O_0(pd) \notin S'$. Hence, we obtain a contradiction and we must have $O_0(pd) \in S$. □

Proposition 3.4.9. *If S is a state reachable from S_1 such that $O_0(pd) \in S$ or $O_1(pd, me_1) \in S$ or $O_2(pd, me_1, me_2) \in S$ or $O_3(pd, me_1, me_2, me_3) \in S$, then an abort request for pd is not on the O-T channel in S,* i.e., *$R_n(k_o, k_t, ab\_req) \notin S$.*

Proof. Intuitively, by proposition 3.2.6, S does not contain $O_{ab?}(pd,-)$ or $O_{ab1}(pd, -)$ or $O_{res1}(pd,-)$. Hence O has still not requested T to abort. The O-T channel is write-protected, and therefore the intruder cannot put this request on the O-T channel.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. Since the globally unique identifier n is freshly generated in this transition, S_0 does not contain an abort request for pd. The rule RG does not create any R_n facts, and hence S_0 also does not contain an abort request for pd on the O-T channel.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps such that $O_0(pd) \in S'$ or $O_1(pd, me_1) \in S'$ or $O_2(pd, me_1, me_2) \in S'$ or $O_3(pd, me_1, me_2, me_3) \in S'$. Then, there is a state S, a transition rule t, and a ground substituion σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.8, $O_0(pd) \in S$ or $O_1(pd, m_1) \in S$ or $O_2(pd, me_1, me_2) \in S$ or $O_3(pd, me_1, me_2, me_3) \in S$.

By induction hypothesis, the abort request for pd is not on the O-T channel in S, *i.e.*, $R_n(k_o, k_t, ab\_req) \notin S$. The rules that can create a R_n fact are: $O_{ab?}$, $O_{res?}$, $R_{res?}$, all rules in $\mathbf{T}$, and the intruder rules SND_R, and SND_T. However, the only rules that can create $R_n(k_o, k_t, ab\_req)$, are $O_{ab?}$, SND_R and SND_T. By proposition 3.1.3, the private keys, k_{os} and k_{ts} are not in the view of the intruder at S'. Hence, the only rule that can create an abort request is $O_{ab?}$.

Therefore, if S contains the abort request for pd on the O-T channel, then t must be $O_{ab?}$. An inspection of the rule tells that S must contain $O_1(pd, me_1)$ and S' must contain $O_{ab?}(pd, -)$. By proposition 3.2.6, S' does not contain $O_0(pd)$ or $O_1(pd, m_1)$ or $O_2(pd, me_1, me_2)$ or $O_3(pd, me_1, me_2, me_3)$, which contradicts our assumption.

Hence, by induction S' does not contain an abort request for pd. □

Proposition 3.4.10. *If S is a state reachable from S_1 such that $O_0(pd) \in S$ or $O_1(pd, me_1) \in S$ or $O_2(pd, me_1, me_2) \in S$ or $O_3(pd, me_1, me_2, me_3) \in S$ then a resolve request for pd is not on the O-T channel in S,* i.e., *$R_n(k_o, k_t, res\_req) \notin S$.*

Proof. Intuitively, by proposition 3.2.6, S does not contain $O_{res?}(pd, -)$ or $O_{ab2}(pd, -)$ or $O_{res2}(pd, -)$. Hence O has still not requested T to resolve. The O-T channel is write-protected, and therefore the intruder cannot put this request on the O-T channel.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. Since the identifier n is freshly generated in this transition, S_0 does not contain an abort request for pd. The rule RG does not create any R_n facts, and hence S_0 also does not contain an resolve request for pd on the O-T channel.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l + 1$ steps such that $O_0(pd) \in S'$ or $O_1(pd, me_1) \in S'$ or $O_2(pd, me_1, me_2) \in S'$ or $O_3(pd, me_1, me_2, me_3) \in S'$. Then there is a state S, a transition rule t, and a ground substituion σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.8, $O_0(pd) \in S$ or $O_1(pd, me_1) \in S$ or $O_2(pd, me_1, me_2) \in S$ or $O_3(pd, me_1, me_2, me_3) \in S$.

By induction hypothesis, the resolve request for pd is not on the O-T channel in S, *i.e.*, $R_n(k_o, k_t, res\_req) \notin S$. The rules that create a R_n fact are: $O_{ab?}$, $O_{res?}$, $R_{res?}$, all rules in $\mathbf{T}$, and the intruder rules SND_R, and SND_T. However, the only rules that can create $R_n(k_o, k_t, res\_req)$, are $O_{res?}$, $R_{res?}$, SND_R and SND_T. By proposition 3.1.3, the private keys, k_{os} and k_{ts} are not in the view of the intruder at S. Hence, SND_R, and SND_T cannot be used to create $R_n(k_o, k_t, res\_req)$.

If the rule $R_{res?}$ creates a fact $R_n(k_o, k_t, mr')$, then mr' must be a pair:

$< PCS(k_1, k_o, k_t, pd'), PCS(k_o, k_1, k_t, pd') >$, where

a) k_1 is a constant of the sort *public_key*, and

b) $pd' =< k_1, k_o, k_t, m', n' >$ for some m' of the sort *preagreed_text*, and n' of the sort *unique_identifier*.

Since k_o and k_r are different constants, $pd =< k_o, k_r, k_t, m, n >$ is different from pd'. Hence $R_{res?}$ cannot create $R_n(k_o, k_t, res\_req)$, the resolve request for pd on the O-T channel. Therefore the only rule that can do so is $O_{res?}$.

Therefore, if S' contains the resolve request for pd on the O-T channel, then t must be $O_{res?}$. An inspection of the rule tells that S must contain $O_3(pd, -)$ and S' contains $O_{res?}(pd, -)$. By proposition 3.2.6, S' cannot contain $O_0(pd)$ or $O_1(pd, me_1)$ or $O_2(pd, me_1, me_2)$ or $O_3(pd, me_1, me_2, me_3)$, which contradicts our assumption.

Hence, S' does not contain a resolve request for pd on the O-T channel and our lemma is true by induction. □

Proposition 3.4.11. *If a state S reachable from S_1 contains $O_0(pd)$ or $O_1(pd, me_1)$ or $O_2(pd, me_1, me_2)$ or $O_3(pd, me_1, me_2, me_3)$ then T has yet to answer a request for pd on the O-T channel in S.*

Proof. Intuitively, by propositions 3.4.9 and 3.4.10, as long as O remains in role states O_0, O_1, O_2, O_3, no abort request or resolve request for pd is deposited on the O-T channel. Hence T is yet to answer a request for pd on the O-T channel.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. S_1 contains $T_0(pd)$ and hence T is yet to answer a request on the O-T channel in S.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps such that $O_0(pd) \in S'$ or $O_1(pd, me_1) \in S'$ or $O_2(pd, me_1, me_2) \in S'$ or $O_3(pd, me_1, me_2, me_3) \in S'$. Then there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.8, $O_0(pd) \in S$ or $O_1(pd, me_1) \in S$ or $O_2(pd, me_1, me_2) \in S$ or $O_3(pd, me_1, me_2, me_3) \in S$.

By induction hypothesis, T is yet to answer a request for *pd* on the *O*-*T* channel in S. Hence S contains $T_0(pd)$ or $T_{rr}(pd, resolved, res\_cn)$. Now, the only way these facts can be consumed by the transition to S is if the transition rule t is one of the following: T_{ab}, T_{or}, T_{rrf1} and T_{rrf2}. In order for these rules to be applied, S must contain either an abort request for *pd* or a resolve request for *pd* on the *O*-*T* channel. By propositions 3.4.9 and 3.4.10, S does not contain either of these. Hence, S' also contains $T_0(pd)$ or $T_{rr}(pd, resolved, res\_cn)$.

Therefore, T is yet to answer a request for *pd* on the *O*-*T* channel in S' and by induction, the statement of our lemma is true. $\square$

Now, we are ready to show lemma 3.2.12.

Lemma 3.2.12. *If in a state S reachable from S_1, O is in a state in which it has requested for an abort_token from T for pd and is waiting for a reply,* i.e., *$O_{ab?}(pd, -) \in S$, then*

<u>*either*</u> *T has answered a request for pd on the O-T channel, and an abort_token or a resolution for pd is on the O-T channel,*

<u>*or*</u> *T has yet to answer a request for pd on the O-T channel, and the abort request*

for pd is on the O-T channel.

Proof. Intuitively, since the *O-T* channel is transparent, an abort request on the channel can be removed only when T acts on it. Once T acts on the request, again by transparency, the answer remains there as long as O does not read it.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. S_1 contains $O_0(pd)$, and therefore by proposition 3.2.6, S_1 does not contain $O_{ab?}(pd, -)$. Hence the statement of the lemma is vacuously true.

Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps such that $O_{ab?}(pd, me_1, ab\_req) \in S'$. Then, there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.8, $O_1(pd, me_1) \in S$ or $O_{ab?}(pd, me_1, ab\_req) \in S$.

We shall consider two cases.

Case 1: $O_1(pd, me_1) \in S$.

Now by proposition 3.2.6, S does not contain $O_{ab?}(pd, me_1, ab\_req)$. Since S' contains $O_{ab?}(pd, me_1, ab\_req)$, the transition t used to obtain S' from S must be

$O_{ab?}$, *i.e.*, the transition must be O sending an abort request for pd. An inspection of the rule tells that an abort request for pd is on the O-T channel.

By proposition 3.4.11, T is yet to answer a request for pd on the O-T channel in S, *i.e.*, S contains $T_0(pd)$ or $T_{rr}(pd, res\_cn)$. Since the transition $O_{ab?}$ does not consume any T_0 or T_{rr} facts, T is yet to answer a request for pd on the O-T channel in S' also.

Hence, in this case, T is yet to answer a request on the O-T channel and the abort request on the O-T channel is still there.

Case 2: $O_{ab?}(pd, me_1, ab\_req) \in S$.

By induction hypothesis, exactly one of the following must be true:

a) T has answered a request for pd on the O-T channel and an abort_token or a resolution for pd is on the O-T channel in S.

b) T is yet to answer a request for pd on the O-T channel and an abort request for pd is on the O-T channel in S.

If a) is true, then by definition S contains one of the following:

$T_{ab}(pd, aborted, ab\_tok)$, $T_{abf}(pd, aborted, ab\_tok)$, $T_{or}(pd, resolved, res\_cn)$, $T_{orf}(pd, resolved, res\_cn)$, $T_{rrf1}(pd, resolved, res\_cn)$, $T_{rrf2}(pd, resolved, res\_cn)$.

An inspection of rules tells that, S' must also contain one of the above and hence T has answered a request for pd on the O-T channel in S also. If S contains an abort_token for pd, then an inspection of the rules tells us that the only way this abort_token can possibly be consumed is if t is one of the following rules: O_{ab1},

O_{ab2}. (We can rule out R_{ab} in a fashion similar to the one we used to rule out $R_{res?}$ in proposition 3.4.10.) If the abort_token for *pd* is consumed by the use of O_{ab1} or O_{ab2}, then S' must contain $O_{ab1}(pd, -)$ or $O_{ab2}(pd, -)$. By proposition 3.2.6, S' cannot contain $O_{ab?}(pd, -)$. This is a contradiction to the assumption that S' contains $O_{ab?}(pd, -)$ and hence the abort_token is on the *O*-*T* channel in S' also. Similarly, if S contains a resolution for *pd* on the *O*-*T* channel, then S' contains a resolution for *pd* on the *O*-*T* channel.

If b) is true, then by arguments similar to case a) above, the only way the abort request for *pd* on the *O*-*T* channel in S, can be removed is when T answers the request. If T answers a request for *pd* on the *O*-*T* channel in the transition to S', then there is an abort_token or a resolution for *pd* on the *O*-*T* channel. Otherwise, the abort request remains on the *O*-*T* channel.

Hence, by induction, the statement of the lemma is true. □

Now we show lemma 3.2.13.

Lemma 3.2.13. *If in a state S reachable from S_1, O is in a state in which it has requested for a resolution from T for pd and is waiting for a reply,* i.e., $O_{res?}(pd, -) \in S$, *then*

<u>either</u> *T has answered a request for pd on the O-T channel and an abort_token or a resolution is on the O-T channel.*

<u>or</u> *T has yet to answer a request for pd on the O-T channel and the request is on the O-T channel.*

Proof. Intuitively, since the *O-T* channel is transparent, a resolve request on the channel can be removed only when T acts on it. Once T acts on the request, again by transparency, the answer remains there as long as O does not read it.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. S_1 contains $T_0(pd)$, and therefore by 3.2.6, S_1 does not contain $O_{res?}(pd, -)$. Hence the statement of the lemma is vacuously true.

Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps such that $O_{res?}(pd, me_1, me_2, me_3, res\_req) \in S'$. Then there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.8, $O_3(pd, me_1, me_2, me_3) \in S$ or $O_{res?}(pd, me_1, me_2, me_3, res\_req) \in S$.

We shall now consider the two cases.

Case 1: $O_3(pd, me_1, me_2, me_3) \in S$.

Now by proposition 3.2.6, S does not contain $O_{res?}(pd, me_1, me_2, me_3, res\_req)$. Since S' contains $O_{res?}(pd, me_1, me_2, me_3, res\_req)$, the transition t used to obtain

S' from S must be $O_{res?}$, *i.e.*, the transition must be O sending a resolve request for *pd*. An inspection of the rule tells that a resolve request for *pd* is on the O-T channel.

Also proposition 3.4.11, T is yet to answer a request for *pd* on the O-T channel in S, *i.e.*, S contains $T_0(pd)$ or $T_{rr}(pd, res\_cn)$. Since the transition $O_{res?}$ does not consume any T_0 or T_{rr} facts, T is yet to answer a request for *pd* on the O-T channel in S' also. Hence, in this case the statement of the lemma is true.

Case 2: $O_{res?}(pd, me_1, me_2, me_3, res\_req) \in S$.

By induction hypothesis, exactly one of the following must be true:

a) T has answered a request for *pd* on the O-T channel and an abort_token or a resolution for *pd* is on the O-T channel in S.

b) T is yet to answer a request for *pd* on the O-T channel and a resolve request for *pd* is on the O-T channel in S.

If a) is true, an inspection of the rules tells that since T has answered a request for *pd* in S, it also has answered a request for *pd* on the O-T channel in S' If S contains an abort_token, then an inspection of the rules tells us that the only way this abort_token can possibly be consumed is if t is one of the following rules: O_{ab1}, O_{ab2}. (We can rule out R_{ab} in a fashion similar to the one we used to rule out $R_{res?}$ in proposition 3.4.10.) If the abort_token for *pd* is consumed by the use of O_{ab1} or O_{ab2}, then S' must contain $O_{ab1}(pd, -)$ or $O_{ab2}(pd, -)$. By proposition 3.2.6, S' cannot contain $O_{res?}(pd, -)$. This is a contradiction to the assumption that S'

contains $O_{res?}(pd, -)$ and hence the abort_token is on the O-T channel. Similarly, if S contains a resolution for T, then S' contains a resolution for T. Hence, the statement of the lemma is true if a) is true.

If b) is true, then by arguments similar to case a) above, the only way a resolve request for pd on the O-T channel can be removed in the transition to T is when T answers the request. In that case, an abort_token or a resolution is deposited on the O-T channel in S'. Otherwise, the resolve request remains on the O-T channel. Hence, the statement of the lemma is true if b) is true.

Therefore by induction, the lemma is true. □

3.4.4 Fairness for honest O

Now, we prove the lemmas needed to show fairness for honest O.

Proposition 3.4.12. *Let S, S' be reachable states such that*

1) S' is obtained from S by a single transition rule.

2) O does not have an abort_token for pd in S.

3) O has an abort_token for pd in S.

Then an abort_token is on the O-T channel in S, i.e.*, S contains $R_n(k_o, k_t, ab\_tok)$.*

Proof. We have O does not have an abort token for pd in S, *i.e.*, S does not contain $O_{ab1}(pd, -)$ or $O_{ab2}(pd, -)$. However, in S', O does have an abort_token for pd. Hence either O_{ab1} or O_{ab_2} is used for the transition from S to S'. An inspection of the rules yields that in order for these rules to apply in S, S must contain

$R_n(k_o, k_t, ab\_tok)$. □

Now we are ready to show lemma 3.2.16.

Lemma 3.2.16. *For all states S reachable from S_1,*

1. *If O has an abort_token for pd then T has an abort_token for pd. Furthermore, in all configurations S' reachable from S, a resolution for pd is not in the view of the intruder in S'.*

2. *If a resolution for pd is in the view of the intruder in S then for all states S' reachable from S, O does not have an abort_token.*

Proof. 1. Note that in S_1, O does not have an abort_token for *pd*. Proposition 3.4.12 tells us that the only way O gets an abort_token for *pd* is when O reads it from the O-T channel. If an abort_token is on the O-T channel then it is in the view of the intruder. By lemma 3.2.5, T must have an abort_token for *pd*.

   Using this, an easy induction shows that as a result of database persistence, if O has an abort_token for *pd* in S, then T must also have an abort_token for *pd*. By database persistence, in all states S' reachable from S, T will have an abort_token for *pd* and by database consistency, T does not have a resolution for *pd*.

2. By lemma 3.2.5, if a resolution is in the view of the intruder, then T must have a resolution for *pd* in S. By database persistence, in all states reachable

from S, T has a resolution for *pd* and by database consistency, T does not have an abort_token for *pd* in those states. Hence by part 1, if S' is a state reachable from S, O does not have an abort_token for *pd* in S'.

$\square$

Proposition 3.4.13. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in $\mathbf{R} \cup \mathbf{T}$. If an abort request for pd is not in the view of the intruder at S, then an abort request for pd is not in the view of the intruder at S' also.*

Proof. Intuitively, since the intruder does not learn the private key of O, the only way intruder can get hold of an abort request for *pd* is when O sends it.

We consider the case when the transition rule applied to obtain S' from S is in $\mathbf{R}$. The other case, that is when the transition rule is in $\mathbf{T}$ can be treated similarly. By proposition 3.1.3, the private key k_{os} is not in the view of the intruder at S, S'. Let $A(S)$ and $A(S')$ be the sets of available messages at S and S' respectively. We have by proposition 3.4.1, an abort request for *pd* is in the view of the intruder at S, S' only if it is an element of $analz(A(S)), analz(A(S'))$ respectively. We have since the abort request is not in the view of the intruder at S, the abort request is not in $analz(A(S))$. We shall show by cases on the rule applied to obtain S' from S, that the abort request for *pd* is not in $analz(A(S'))$:

1. Rule R_2: A message $PCS(k_2, pd', k_1, k_3)$ is deposited on the network, where k_1, k_2, k_3 are keys of the sort *public_key*, and $pd' = \langle m', n', k_1, k_2, k_3\rangle$ where m'

and n' are terms of the sort *preagreed_text* and *nonce* respectively. (Note pd' may not be the same as pd: there may be several concurrent executions of the protocol). Now we have by proposition 3.4.1, $analz(A(S')) = analz(A(S)) \cup analz(PCS(k_1, pd', k_2, k_3))$. It can be shown that $analz(PCS(k_1, pd', k_2, k_3)) = \{PCS(k_1, pd', k_2, k_3)\} \cup analz(pd')$. Clearly, $analz(pd')$ does not contain an abort request for pd, and hence $analz(PCS(k_1, pd', k_2, k_3))$ also does not contain an abort request for pd. Since $analz(A(S))$ also does not contain an abort request for pd, we get an abort request for pd is not in $analz(A(S'))$.

2. Rule R_4: A message of the form $sig(k_2, < m', n', k_1, k_2, k_3 >)$ is deposited on the network. Clearly, $analz(sig(k_2, < m', n', k_1, k_2, k_3 >)) = \{sig(k_2, < m', n', k_1, k_2, k_3 >)\} \cup analz(pd')$, and $analz(pd')$ does not contain an abort request for pd. Clearly, $sig(k_2, < m', n', k_1, k_2, k_3 >)$ cannot be the abort request for pd (since the abort request for pd is the signature on the pair $< abort, pd >$). Therefore, since $analz(A(S))$ also does not contain an abort request for pd, abort request for pd and is not in $analz(A(S'))$.

3. Rule $R_{res?}$: A message $< PCS(k_1, pd', k_2, k_3), PCS(k_2, pd', k_1, k_3) >$ is deposited on the channel between k_2 and k_3. As in case 1, once again an abort request for pd is not in $analz(A(S'))$.

4. Rule R_1, R_{quit}, R_2, R_{com}, R_{ab}, R_{res}: The set of available messages actually decreases, and therefore an abort request for pd and is not in $analz(A(S'))$.

Therefore, we get that the view of the intruder at S' does not contain an abort request for pd. $\square$

Proposition 3.4.14. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* $\mathbf{I}$. *If an abort request for pd is not in the view of the intruder at S, then an abort request for pd is not in the view of the intruder at S' also.*

Proof. By proposition 3.1.2, view of the intruder does not change by an application of a rule in $\mathbf{I}$ other than GEN. Now, if GEN is used, then the set of available messages increases by a new constant u of the sort $mssg$. Since $analz(u) = \{u\}$, we get an abort request for *pd* is in the view of the intruder at S' only if it is in the view of the intruder at S. $\square$

Proposition 3.4.15. *Let S, S' be reachable states such that S' is obtained from S by the application of the role generation rule RG. If an abort request for pd is not in the view of the intruder at S, then an abort request for pd is not in the view of the intruder at S' also.*

Proof. If *RG* is used, then the set of available messages increases by a new constant n' of the sort $unique\_identifier$. Since $analz(n') = \{n'\}$, we get an abort request for *pd* is in the view of the intruder at S' only if it is in the view of the intruder at S. $\square$

Proposition 3.4.16. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* $\mathbf{O}$. *If an abort request for pd is not in* *the view of the intruder at S, then it is in the view of the intruder at S' only if $O_{ab?}(pd, -) \in S'$.*

Proof. Intuitively, since the intruder does not learn the private key of O, the only way intruder can get hold of an abort request for pd is when O sends it.

The proof is again by cases on the rule being used in the transition from S to S'. As in the proof of proposition 3.4.13, we can show that if the transition is anything other than $O_{ab?}$, then an abort request is not in the view of the intruder at S' if an abort request is not in the view of the intruder at S. As the key k_{os} is not in the view of the intruder at S, S' (proposition 3.1.3), an abort request is in the view of the intruder in S, S' only if it is contained in $analz(A(S)), analz(A(S'))$ respectively.

If the rule $O_{ab?}$ is used to transit from S to S', then the only new available message at S' is $sig(k_1, < abort, pd' >)$ where $pd' =< m', n', k_1, k_2, k_3 >$ where k_1, k_2, k_3 are terms of sort $public\_key$, m' is a term of the sort $preagreed\_text$ and n' is a term of the sort $unique\_identifier$. In this case, $O_{ab?}(pd', PCS(k_1, pd', k_2, k_3), sig(k_1, < abort, pd' >))$ is contained in S'.

Now it can be easily shown that $analz(sig(k_1, < abort, pd' >))$ is the set that consists of: $sig(k_1, < abort, pd' >), < abort, pd' >, abort, pd', k_1, k_2, m', n'$ and k_3. Clearly, the abort request for pd is in $analz(sig(k_1, < abort, pd' >))$ only if pd' is the same as pd. Hence the abort request is in $analz(S')$ only if it is in $analz(S)$ or if *pd'* is the same as *pd*.

By hypothesis, an abort request for *pd* is not in the view of the intruder at S and hence not in $analz(S)$. Hence if an abort request is in the view of the intruder then *pd'* must be the same as *pd*, and we get $O_{ab?}(pd, -) \in S'$. $\square$

Now, we prove lemma 3.2.17.

Lemma 3.2.17. *For all state S reachable from S_1,*

1. *If $O_i(pd, -)$ in S, where $i \in \{0, 1, 2, 3, res?\}$ then neither an abort request nor an abort_token for pd is in the view of the intruder at S.*
2. *S does not contain $O_{ab2}(pd, -) \notin S$,* i.e.*, the only way O has an abort_token is if it is in the state O_{ab1}.*

Proof. 1. Please note that if an abort_token is in the view of the intruder, then an abort request is in the view of the intruder. Intuitively, by propositions 3.4.13, 3.4.14, 3.4.15, and 3.4.16 if a state transition produces an abort request that was not present before, then O must have sent the abort request.

We prove the lemma by induction on number of steps that it takes to go from S_1 to S.

Base case: S is S_1. Since the identifier n is freshly generated in the transition from S_0 to S_1, clearly an abort request for *pd* is not in the view of the intruder at S_0. Hence by proposition 3.4.15, an abort request *pd* is not in the view of the intruder at S_1 also.

Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps. Consider a state S' such that S' is reachable from S_1 in $l+1$ steps and $O_i(pd, -)$ in S' for $i \in \{0, 1, 2, 3, res?\}$. Since S' is reachable from S_1 in $l+1$ steps, there must be a state S such that:

1) S' is reachable from S by a single transition,

2) S is reachable from S_1 by n steps, and

3) by proposition 3.4.8, $O_i(pd, -)$ in S for $i \in \{0, 1, 2, 3, res?\}$.

By induction hypothesis, an abort request for pd is not in the view of the intruder in S. So, if the abort request for pd is in the view of the intruder in S' , then by propositions 3.4.13, 3.4.14, and 3.4.15, and 3.4.16 $O_{ab?}(pd, -) \in S'$. By proposition 3.2.6, S' cannot contain $O_i(pd, -)$ for $i \in \{0, 1, 2, 3, res?\}$. Hence, we obtain a contradiction and therefore an abort request is not in the view of the intruder in S' also.

2. We prove this by induction on the number of steps it takes to obtain S from S_1.

   Base Case: S is S_1: We have $O_0(pd) \in S_1$. By proposition 3.2.6, S_1 does not contain $O_{ab2}(pd, -)$.

   Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps. Consider a state S' such that S' is reachable from S_1 in $l+1$ steps. Since S' is reachable from S_1 in $l+1$ steps, there must be a state S such that:

1) S' is reachable from S by a single transition, and

2) S is reachable from S_1 by l steps.

By induction hypothesis, S does not contain $O_{ab2}(pd, -)$. If S' contains $O_{ab2}(pd, -)$, then

a) By proposition 3.4.8, S contains $O_{res?}(pd, -)$.

b) An inspection of rules tells that the transition from S to S' uses O_{ab2} and an abort_token for *pd* is in on the *O*-*T* channel in S.

This contradicts part 1 of the lemma since an abort_token for *pd* cannot be in the view of the intruder of S if S contains $O_{res?}(pd, -)$. Hence S' does not contain $O_{ab2}(pd, -)$ also.

□

Proposition 3.4.17. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in $\mathbf{R} \cup \mathbf{T}$. If O's signature on pd,* i.e., *$sig(k_o, pd)$, is not in the view of the intruder at S, then it is also not in the view of the intruder at S'.*

Proof. Intuitively, since the intruder does not learn the private key of O, the only way intruder can get hold of an abort request for *pd* is when O sends it.

We consider the case when the transition rule applied to obtain S' from S is in $\mathbf{R}$. The other case, that is when the transition rule is in $\mathbf{T}$ can be treated similarly. By proposition 3.1.3, the private key k_{os} is not in the view of the intruder at S, S'. Let $A(S)$ and $A(S')$ be the sets of available messages at S and S' respectively. We have by proposition 3.4.1, O's signature on pd is in the view of the intruder at S, S' only if it is an element of $analz(A(S)), analz(A(S'))$ respectively. We have since O's signature on pd is not in the view of the intruder at S, it is not in $analz(A(S))$. We shall show by cases on the rule applied to obtain S' from S, that O's signature on pd is not in $analz(A(S'))$:

1. Rule R_2: A message $PCS(k_2, pd', k_1, k_3)$ is deposited on the network, where k_1, k_2, k_3 are keys of the sort public_key, and $pd' = \langle m', n', k_1, k_2, k_3 \rangle$ where m' and n' are terms of the sort *contract_text* and nonce respectively. (Note pd' may not be the same as pd: there may be several concurrent executions of the protocol). Now we have by proposition 3.4.1, $analz(A(S')) = analz(A(S)) \cup analz(PCS(k_1, pd', k_2, k_3))$. Now, $analz(PCS(k_1, pd', k_2, k_3)) = \{PCS(k_1, pd', k_2, k_3)\} \cup analz(pd')$. Clearly $analz(pd')$ does not contain O's signature on pd and $PCS(k_1, pd', k_2, k_3)$ is also not O's signature on pd. Since $analz(S)$ also does not contain O's signature on pd, we get O's signature on pd is not in $analz(A(S'))$.

2. Rule R_4: A message of $sig(k_2, < m', n', k_1, k_2, k_3 >)$ is deposited on the network. Once again, O's signature on pd is in $analz(S')$ only if $analz(sig(k_2, < m', n', k_1, k_2, k_3 >))$ contains $sig(k_o, pd)$. Clearly, this will be true only if k_2 is k_o and $pd =< m', n', k_1, k_2, k_3 >$. But pd is $< m, n, k_o, k_r, k_t >$. Hence k_2 must be k_r. This would mean that k_r and k_o are same. A contradiction, since k_r and k_o are different. Hence O's signature on pd is not in $analz(A(S'))$.

3. Rule $R_{res?}$: A message $< PCS(k_1, pd', k_2, k_3), PCS(k_2, pd', k_1, k_3) >$ is deposited on the channel between k_2 and k_3. As in case 1, once again O's signature on pd is not in $analz(A(S'))$.

4. Rule R_1, R_{quit}, R_2, R_{com}, R_{ab}, R_{res}: The set of available messages actually decreases, hence O's signature on pd is not in $analz(A(S'))$.

$\square$

Proposition 3.4.18. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* **I**. *If O's signature on pd is not in the view of the intruder at S, then O's signature on for pd is not in the view of the intruder at S, then an abort request for pd is not in the view of the intruder at S' also.*

Proof. By proposition 3.1.2, view of the intruder does not change by an application of a rule in **I** other than GEN. Now, if GEN is used, then the set of available messages increases by a new constant u of the sort $mssg$. Since $Analz(u) = \{u\}$, the result follows. $\square$

Proposition 3.4.19. *Let S, S' be reachable states such that S' is obtained from S by the application of the role generation rule RG. If O's signature on pd is not in the view of the intruder at S, then O's signature on pd is not in the view of the intruder at S' also.*

Proof. If RG is used, then the set of available messages increases by a new constant n' of the sort $unique\_identifier$. Since $Analz(n) = \{n'\}$, the result follows. $\square$

Proposition 3.4.20. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* $\mathbf{O}$. *If O's signature on pd is not in the view of the intruder at S, it in the view of the intruder at S' only if $O_3(pd, -) \in S'$.*

Proof. Intuitively, since the intruder does not learn the private key of O, the only way intruder can get hold of O's signature on *pd* is when O sends it.

The proof is again by cases on the rule being used in the transition from S to S'. As in the proof of proposition 3.4.17, we can show that if the transition is anything other than O_3, then O's signature on *pd* is not in the view of S' if it is not in the view of the intruder in S.

As the key k_{os} is not in the view of the intruder at S, S' (proposition 3.1.3), O's signature is in the view of the intruder in S, S' only if can be obtained by analyzing the messages available to the intruder in S, S', *i.e.*, only if it is contained in $analz(A(S)), analz(A(S'))$ respectively.

If the rule O_3 is used to transit from S to S', then the only new available message at S' is $sig(k_1, pd')$ where $pd' =< m', n', k_1, k_2, k_3 >$ where k_1, k_2, k_3 are terms of sort public_key, m' is a term of the sort *preagreed_text* and n' is a term of the sort *nonce*. In this case, $O_3(pd', sig(k_1, pd))$ is contained in S'.

Now it can be easily shown that $Analz(sig(k_1, pd'))$ is the set that consists of: $sig(k_1, pd'), pd', k_1, k_2, m', n'$ and k_3. Clearly, O's signature for *pd* is in $analz(A(S'))$ only if it is in $analz(S)$ or if *pd'* is the same as *pd*. By hypothesis, O's signature on *pd* is not in the view of the intruder at S. Hence, it is not in $analz(S)$. Therefore if it is in $analz(A(S'))$, then pd' is the same as pd, and we get $O_3(pd, -) \in S'$ $\square$

Finally, we show lemma 3.2.18.

Lemma 3.2.18. *For all states S reachable from S_1,*

1. *If S contains, $O_i(pd, -)$ for $i \in \{0, 1, ab?, ab1, res1\}$, then O's signature on pd, $sig(k_o, pd)$, in not in the view of the intruder.*

2. *If O has an abort_token for pd then for all S' reachable from S, O's signature on pd, $sig(k_o, pd)$, is not in the view of the intruder. If O's signature on pd is in the view of the intruder then for all S' reachable from S, O does not have an abort_token in S'.*

Proof. 1. Intuitively, by propositions 3.4.17, 3.4.18, 3.4.19 and 3.4.20 if a state transition produces O's signature on pd that was not present before, then O must have sent it.

We prove this by induction on number of steps that it takes to go from S_1 to S.

Base case: S is S_1. Since the identifier n is freshly generated in the transition from S_0 to S_1, clearly O's signature on pd is not in the view of the intruder at S_0. Hence by proposition 3.4.19, $sig(k_o, pd)$ is not in the view of the intruder at S_1 also.

Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps. Consider a state S' such that S' is reachable from S_1 in $l+1$ steps and $O_i(pd, -)$ in S for $i \in \{0, 1, ab?, ab1, res1\}$. Since S' is reachable from S_1 in $l+1$ steps, there must be a state S such that:

1) S' is reachable from S by a single transition,

2) S is reachable from S_1 by l steps, and

3) by proposition 3.4.8, $O_i(pd, -)$ in S for $i \in \{0, 1, ab?, ab1, res1\}$.

By induction hypothesis, O's signature on *pd* is not in the view of the intruder in S also. So, if it is in the view of the intruder in S , then by propositions 3.4.17, 3.4.18, 3.4.19 and 3.4.20, $O_3(pd, -) \in S'$. By proposition 3.2.6, S' cannot contain $O_i(pd, -)$ for $i \in \{0, 1, ab?, ab1, res1\}$. A contradiction and hence O's signature on *pd* is not in the view of the intruder in S' also.

2. If O has an abort_token for *pd* at S, then by proposition 3.4.8, O has the abort_token for all states S' reachable from S. By lemma 3.2.17, if O has abort token in S', then $O_{ab1}(pd, -) \in S'$. By part 1 of the lemma, O' signature on *pd* is not in the view of the intruder.

   If O's signature is in the view of the intruder at S, then by part 1 and proposition 3.2.6, S contains $O_i(pd, -)$ for $i \in \{2, 3, com, res?, res2, ab2\}$. By proposition 3.4.8, if S' is reachable from S, then S' contains $O_j(pd, -)$ for $j \in \{2, 3, com, res?, res2, ab2\}$. By lemma 3.2.17, S' cannot contain $O_{ab2}(pd, -)$, and hence must contain $O_j(pd, -)$ for $j \in \{2, 3, com, res?, res2\}$. Therefore, O does not have an abort_token for *pd* in S'.

□

3.4.5 Effectiveness for Honest R

Now we proof lemma 3.2.27 needed in the proof of effectiveness of honest R. We start by proving proposition 3.2.21

Proposition 3.2.21. *If S is a state reachable from S_1, then S contains exactly one of the following:*

$R_0(pd)$, $R_{quit}(pd)$, $R_1(pd, me_1)$, $R_2(pd, me_1, me_2)$, $R_3(pd, me_1, me_2, me_3)$,

$R_{com}(pd, me_1, me_2, me_3, me_4)$, $R_{res?}(pd, me_1, me_2, res\_req)$,

$R_{ab}(pd, me_1, me_2, ab\_req, ab\_tok)$, $R_{res}(pd, me_1, me_2, res\_req, res\_cn)$.

Furthermore, there is exactly one occurrence of the fact $R_i(pd, -)$ for each $i \in \{0, 1, 2, 3, quit, ab?, res?, ab, res, com\}$.

Proof. Intuitively, the freshness of the globally unique identifier guarantees that there is only one instance of a fact of the form $R_i(pd, -)$. Furthermore, the rules in the protocol theory **R** ensure that whenever one fact $R_i(pd, -)$ is consumed, another fact $R_j(pd, -)$ is created.

We prove the proposition by induction on the length of reachability of S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. Since the identifier n is freshly generated in this transition, S_0 does not contain any fact of the form $R_i(pd, -)$ for $i \in \{0, 1, 2, 3, quit, ab?, res?, ab, res, com\}$. In the transition $R_0(pd)$ is created. No other fact of the form $R_i(pd, -)$ is created. Hence, the statement of the proposition is true.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps. Then, there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S is by the application of t using σ.

2) S is reachable from S_1 by l steps.

By Induction hypothesis, S contains exactly one fact of the following:
$R_0(pd)$, $R_{quit}(pd)$, $R_1(pd, me_1)$, $R_2(pd, me_1, me_2)$, $R_3(pd, me_1, me_2, me_3)$,
$R_{com}(pd, me_1, me_2, me_3, me_4)$, $R_{res?}(pd, me_1, me_2, res\_req)$,
$R_{ab}(pd, me_1, me_2, ab\_req, ab\_tok)$, $R_{res}(pd, me_1, me_2, res\_req, res\_cn)$.
Furthermore, there is exactly one occurrence of the fact $R_i(pd, -)$ for each
$i \in \{0, 1, 2, 3, quit, ab?, res?, ab, res, com\}$.

We shall show that if S contains $R_0(pd)$, then the statement of the proposition holds in S' also. The other cases can be treated similarly. Now, if the transition from S to S' uses a transition in $\mathbf{T} \cup \mathbf{O} \cup \mathbf{I}$, then since no R_i fact is created or consumed in this transition, the statement of the proposition still holds in S'.

If the transition from S to S' uses the transition RG, then a fact $R_0(pd')$ is created, where $pd' =< k_1, m_1, n', k_2, k_t >$, where n' is freshly generated and is different from the globally unique identifier in pd. Hence, pd and pd' are different. No other O_i fact is created or consumed. Hence, the statement of the proposition is true in S'.

If the transition from S to S' uses the transition R_1, then a fact $R_0(pd')$ is consumed and a fact of the form $R_1(pd', -)$ is created. No other R_i fact is created or consumed. Now, there are two cases: pd' is the same as pd, or pd is different from pd'. If pd is the same as pd, then $R_0(pd)$ is consumed and $R_1(pd, me_1)$ is created, and the statement of the proposition is true. If pd is different from pd', then no fact of the form $R_i(pd, -)$ is created or consumed in the transition and the proposition is still true.

If the transition from S to S' uses the transition R_{quit}, then a fact $R_0(pd')$ is consumed and a fact of the form $R_{quit}(pd'$ is created. No other R_i fact is created or consumed. Now, there are two cases: pd' is the same as pd, or pd is different from pd'. If pd is the same as pd, then $R_0(pd)$ is consumed and $R_{quit}(pd)$ is created, and the statement of the proposition is true. If pd is different from pd', then no fact of the form $R_i(pd, -)$ is created or consumed in the transition and the proposition is still true.

If the transition from S to S' uses the transition R_2, then then a fact $R_1(pd', -)$ is consumed and a fact of the form $R_2(pd', -)$ is created. No other fact R_i is created or consumed. Since S contains $R_0(pd)$, S' also contains $R_0(pd)$. By induction hypothesis, there is only one occurrence of the form $R_i(pd, -)$. Hence S the only fact of the form contained by S is $R_0(pd)$.

In particular S cannot contain $R_1(pd, -)$. Therefore, pd must be different from pd', and hence the transition to S' does not produce a fact of the form $R_i(pd, -)$. Hence the statement of the proposition is true in S'. Similarly, if the transition from S' to S uses one of the transitions: R_2, R_3, R_{com}, $R_{res?}$, R_{ab} or R_{res}, then the statement of the proposition is true in S. Therefore if S contains $R_0(pd, -)$, the statement of the proposition is true in S'.

Hence, by induction, the proposition is true. □

Proposition 3.4.21. *Let S and S' be states reachable from S_1 such that S' is obtained from S by an application of a single rule. Then*

1. *If $R_0(pd) \in S$, then $R_0(pd) \in S'$ or $R_1(pd, -) \in S'$ or $R_{quit}(pd, -) \in S'$. If $R_0(pd) \in S'$, then $R_0(pd) \in S$.*

2. *If $R_1(pd, -) \in S$, then $R_1(pd, -) \in S'$ or $R_2(pd, -) \in S'$. If $R_1(pd, -) \in S'$, then $R_0(pd) \in S$ or $R_1(pd, -) \in S$.*

3. *If $R_2(pd, -) \in S$, then $R_2(pd, -) \in S'$ or $R_{res?}(pd, -) \in S'$ or $R_3(pd, -) \in S'$. If $R_2(pd, -) \in S'$, then $R_1(pd, -) \in S$ or $R_2(pd, -) \in S$.*

4. *If $R_3(pd, -) \in S$, then $R_3(pd, -) \in S'$ or $R_{com}(pd, -) \in S'$. If $R_3(pd, -) \in S'$, then $R_2(pd) \in S$ or $R_3(pd, -) \in S$.*

5. *If $R_{com}(pd, -) \in S$, then $R_{com}(pd, -) \in S'$. If $R_{com}(pd, -) \in S'$, then $R_3(pd, -) \in S$ or $R_{com}(pd, -) \in S$.*

6. *If $R_{quit}(pd, -) \in S$, then $R_{quit}(pd, -) \in S'$. If $R_{quit}(pd, -) \in S'$ then $R_o(pd) \in S$ or $R_{quit}(pd, -) \in S$.*

7. *If $R_{res?}(pd,-) \in S$, then $R_{res?}(pd,-) \in S'$ or $R_{ab}(pd) \in S'$ or $R_{res}(pd,-) \in S'$. If $R_{res?}(pd,-) \in S'$ then $R_2(pd,-) \in S$ or $R_{res?}(pd,-) \in S$.*

8. *If $R_{ab}(pd,-) \in S$, then $R_{ab}(pd,-) \in S'$. If $R_{ab}(pd,-) \in S'$ then $R_{res?}(pd,-) \in S$ or $R_{ab}(pd,-) \in S$.*

9. *If $R_{res}(pd,-) \in S$, then $R_{res}(pd,-) \in S'$. If $R_{res}(pd,-) \in S'$ then $R_{res?}(pd,-) \in S$ or $R_{res}(pd,-) \in S$.*

Proof. We shall show part 1. The other parts can be similarly proved. If $R_0(pd) \in S$, then an inspection of rules tells that the only way $R_0(pd)$ can be consumed in the transition to S' is if the rule R_1 or the rule R_{quit} is used. Now, if R_1 is used and $R_0(pd,-)$ consumed in the transition, then an inspection of the rule tells that a fact $R_1(pd,-)$ is created. Now, if R_{quit} is used and $R_0(pd,-)$ consumed in the transition, then an inspection of the rule tells that a fact $R_{quit}(pd)$ is created. Hence, if $R_0(pd) \in S$, then $R_0(pd) \in S'$ or $R_{quit}(pd)$ or $R_1(pd,-) \in S'$.

Now suppose $R_0(pd) \in S'$ and $R_0(pd) \notin S$. Then, since an R_0 fact is created in the transition to S', an inspection of the rules tells that RG must be used in the transition from S to S'. In this transition, $R_0(pd')$ is created. Now, in the transition the identifier in pd' is freshly generated and must be different from pd. Hence $R_0(pd')$ is different from $R_0(pd)$ and we get $R_0(pd) \notin S'$. Hence, we obtain a contradiction and we must have $R_0(pd) \in S$. $\square$

Proposition 3.4.22. *If S is a state reachable from S such that $R_0(pd) \in S$ or*

$R_1(pd, me_1) \in S$ or $R_2(pd, me_1, me_2) \in S$ then a resolve request for pd is not on the R-T channel in S, i.e., *$R_n(k_r, k_t, res\_req) \notin S$.*

Proof. Intuitively, by proposition 3.2.21, S does not contain $R_{res?}(pd, -)$ or $R_{ab}(pd, -)$ or $R_{res}(pd, -)$. Hence R has still not requested T to resolve. The R-T channel is write-protected, and therefore the intruder cannot put this request on the R-T channel.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. Since the identifier n is freshly generated in this transition, S_0 does not contain an abort request for *pd*. The rule RG does not create any R_n facts, and hence S_1 also does not contain an resolve request for *pd*.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps such that $R_0(pd) \in S'$ or $R_1(pd, me_1) \in S'$ or $R_2(pd, me_1, me_2) \in S'$. Then there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.21, $R_0(pd) \in S$ or $R_1(pd, me_1) \in S$ or $R_2(pd, me_1, me_2) \in S$.

By induction hypothesis, the resolve request for *pd* is not on the R-T channel in S, *i.e.*, $R_n(k_r, k_t, res\_req) \notin S$. The rules that create a R_n fact are: $O_{ab?}$, $O_{res?}$, $R_{res?}$, all rules in $\mathbf{T}$, and the intruder rules SND_R, and SND_T. However, the only rules that can create $R_n(k_r, k_t, res\_req)$, are $O_{res?}$, $R_{res?}$, SND_R and SND_T. By proposition 3.1.3, the private keys, k_{rs} and k_{ts} are not in the view of the intruder at S. Hence $R_n(k_r, k_t, res\_req)'$ cannot be created by SND_R and SND_T

If the rule $O_{res?}$ creates a fact $R_n(k_r, k_t, mr')$, then mr' must be a pair:
$< PCS(k_r, k_1, k_t, pd'), PCS(k_1, k_r, k_t, pd') >$, where
i) k_1 is a constant of the sort *public_key*, and
ii) $pd' =< k_r, k_1, k_t, m', n' >$ for some m' of the sort *preagreed_text* and n' of the sort $unique\_identifier$.
Since k_o and k_r are different constants, $pd =< k_o, k_r, k_t, m, n >$ is different from pd'. Hence $O_{res?}$ cannot create a resolve request for pd on the *R-T* channel and the only rule that can create a resolve request for pd on *R-T* channel is $R_{res?}$.

Therefore, if S' contains the resolve request for pd on the *R-T* channel, then t must be $R_{res?}$ and and an inspection of the rule $R_{res?}$ tells that S' must contain $R_{res?}(pd, -)$. By proposition 3.2.21, S' cannot contain $R_0(pd)$ or $R_1(pd, me_1)$ or $R_2(pd, me_1, me_2)$ which contradicts our assumption.

Hence, S' does not contain a resolve request for pd on the *R-T* channel and our proposition is true by induction. □

Proposition 3.4.23. *If S is a state reachable from S_1 that contains $R_0(pd)$ or $R_1(pd, me_1)$ or $R_2(pd, me_1, me_2)$, then T has yet to answer a request for pd on the*

R-T channel in S_1.

Proof. Intuitively, by proposition 3.4.22, as long as R remains in role states R_0, R_1 and R_2, no resolve request for pd is deposited on the R-T channel. Hence T is yet to answer a request for pd on the R-T channel.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule RG. S_1 contains $T_0(pd)$ and hence T is yet to answer a request on the R-T channel in S.

Induction Hypothesis: Let the statement of the proposition be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps such that $R_0(pd) \in S'$ or $R_1(pd, me_1) \in S'$ or $R_2(pd, me_1, me_2) \in S'$. Then, there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.21, $R_0(pd) \in S$ or $R_1(pd, me_1) \in S$ or $R_2(pd, me_1, me_2) \in S$.

By induction hypothesis, T is yet to answer a request for pd on the R-T channel in S, *i.e.*, S contains $T_0(pd)$ or $T_{ab}(pd, aborted, ab\_tok)$ or $T_{or}(pd, resolved, res\_cn)$. Now, the only way these facts can be consumed by the transition to S is if the transition rule t is one of the following: T_{rr}, T_{abf}, T_{orf}. In order for these rules to be applied, S must contain a resolve request for pd on the R-T channel. By proposition 3.4.22, S does not contain this. Hence, S' also contains $T_0(pd)$ or $T_{ab}(pd, aborted, ab\_tok)$ or $T_{or}(pd, resolved, res\_cn)$.

Therefore, T is yet to answer a request for *pd* on the *R*-*T* channel in S'. □

Now, we are ready to show lemma 3.2.27.

Lemma 3.2.27. *If in a state S reachable from S_1, R is in a state in which it has requested for a resolution from T for pd and is waiting for a reply,* i.e., $R_{res?}(pd-) \in S$*, then*

<u>either</u> T has answered a request for pd on the R-T channel and an abort_token or a resolution for pd is on the R-T channel.

<u>or</u> T has yet to answer a request for pd on the R-T channel and the resolve request is on the R-T channel.

Proof. Intuitively, since the *R*-*T* channel is transparent, a resolve request on the channel can be removed only when T acts on it. Once T acts on the request, again by transparency, the answer remains there as long as R does not read it.

We shall prove the proposition by induction on the number of steps it takes to go from S_1 to S.

Base case: S is S_1. S_1 is obtained from S_0 using the role generation rule *RG*. S_1 contains $R_0(pd)$, and therefore by proposition 3.2.21, S_1 does not contain $R_{res?}(pd, -)$. Hence the statement of the lemma is vacuously true.

Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps.

Let S' be a state reachable from S_1 by $l+1$ steps such that $R_{res?}(pd, me_1, me_2, res\_req) \in S'$. Then there is a state S, a transition rule t, and a ground substitution σ such that

1) S' is obtained from S by the application of t using σ.

2) S is reachable from S_1 by l steps.

By proposition 3.4.21, $R_2(pd, me_1, me_2) \in S$ or $R_{res?}(pd, me_1, me_2, res\_req) \in S$.

We shall consider two cases.

Case 1: $R_2(pd, me_1, me_2) \in S$.

Now by proposition 3.2.21, S does not contain $R_{res?}(pd, me_1, me_2, res\_req)$. Since S' contains $R_{res?}(pd, -)$, the transition t used to obtain S' from S must be $R_{res?}$, *i.e.*, the transition must be R sending a resolve request for *pd*. Therefore, a resolve request for *pd* is on the *R*-*T* channel.

By proposition 3.4.23, T is yet to answer a request for *pd* on the *R*-*T* channel in S, *i.e.*, S contains $T_0(pd)$ or $T_{ab}(pd, -)$ or $T_{or}(pd, -)$. Since the transition $R_{res?}$ does not consume any T_0 or T_{ab} or T_{or} facts, T is yet to answer a request for *pd* on the *R*-*T* channel in S' also. Hence the statement of the lemma is true in this case.

Case 2: $R_{res?}(pd, me_1, me_2, res\_req) \in S$.

By induction hypothesis, exactly one of the following must be true:

i) T has answered a request for *pd* on the *R*-*T* channel and an abort_token or a resolution for *pd* is on the *R*-*T* channel in S.

ii) T is yet to answer a request for *pd* on the *R*-*T* channel and a resolve request for

pd is on the R-T channel in S.

If i) is true, then an inspection of rules tell that, T has answered a request for pd on the R-T channel in S' also. If S contains an abort_token on R-T channel, then an inspection of the rules tells us that the only way this abort_token can possibly be consumed is if t is R_{ab}. (We can rule out O_{ab1} or O_{ab2} in a fashion similar to the one we used to rule out $O_{res?}$ in proposition 3.4.22.) If the abort_token for pd is consumed by the use of R_{ab} then S' must contain $R_{ab}(pd, -)$. By proposition 3.2.21, S' cannot contain $R_{res?}(pd, -)$. This is a contradiction to the assumption that S' contains $R_{res?}(pd, -)$ and hence the abort_token is on the R-T channel in S'. Similarly, if S contains a resolution for pd on the R-T, then S' contains a resolution for pd on the R-T channel.

If ii) is true, then an inspection of the rules tells that the only way a resolve request can be removed in the transition to S' is if T answers the request. In that case, an abort_token or a resolution for pd is deposited on the R-T channel in S. Otherwise, the resolve request remains on the R-T channel. $\square$

3.4.6 Fairness for honest R

Proposition 3.4.24. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in $\mathbf{O} \cup \mathbf{T}$. If,*

1. *me_2, i.e., $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S,*

2. *$O_i(pd, -) \notin S$ for all $i \in \{2, 3, com, res?, res1, res2\}$, and*

3. $T_0(pd)$ or $T_{ab}(pd, -) \in S$.

Then,

1. me_2, i.e., *$PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S',*

2. $O_i(pd, -) \notin S'$ for all $i \in \{2, 3, com, res?, res1, res2\}$, and

3. $T_0(pd)$ or $T_{ab}(pd, -) \in S'$.

Proof. Intuitively, R's private key is not known to the intruder. Hence, the intruder cannot create $PCS(k_r, pd, k_o, k_t)$, even if it can create its simulation $FakeSign$. Therefore if $PCS(k_r, pd, k_o, k_t)$ is not sent then neither T nor O ever it.

Let $A(S)$ and $A(S')$ be the sets of available messages at S, S' respectively. By proposition 3.1.3, the private keys k_{rs} is not in the view of the intruder at S and S'. By proposition 3.4.1, $PCS(k_r, pd, k_o, k_t)$ is in view of the intruder at S, S' only if it is in $analz(A(S)), analz(A(S'))$ respectively. We have by hypothesis, $PCS(k_r, pd, k_o, k_t)$ is not in $analz(A(S))$.

We shall prove the proposition when the rule used in transition from S to S_1 is O_1 and O_2. The other cases can be treated similarly.

Rule O_1: In this case, a message $PCS(k_1, pd', k_1, k_3)$ is deposited on the network, where k_1, k_2, k_3 are keys of the sort *public_key* and $pd' =< m', n', k_1, k_2, k_3 >$ where m' and n' are terms of the sorts *preagreed_text* and *unique_identifier* respectively. (Note pd' may not be the same as pd: there may be several concurrent executions of the protocol). By proposition 3.4.1, $analz(A(S')) = analz(A(S)) \cup$

$analz(PCS(k_1, pd', k_2, k_3))$. Since, $PCS(k_r, pd, k_o, k_t)$ is not in $analz(A(S))$, it will be in $analz(A(S'))$ only if it is in $analz(PCS(k_1, pd', k_2, k_3))$.

Now, $analz(PCS(k_1, pd', k_2, k_3))= \{PCS(k_1, pd', k_2, k_3)\} \cup analz(pd')$. It can be easily shown that $PCS(k_r, pd, k_o, k_t)$ is not in $analz(pd')$. If $PCS(k_1, pd', k_2, k_3)$ is the same as $PCS(k_r, pd, k_o, k_t)$, then k_1, k_2, and pd must be same as k_r, k_o and pd respectively. We have $pd =< m, n, k_o, k_r, k_t >$ and $pd' =< m', n', k_1, k_2, k_3 >$. Since pd is same as pd', we must get k_1 must be same as k_o. Hence, we get k_0 and k_r must be the same. But they are different, and hence $PCS(k_r, pd, k_o, k_t)$ is not in $analz(A(S'))$. Also, the transition rule O_1 does not create any O_i facts for all $i \in \{2, 3, com, res?, res1, res2\}$, and does not consume any T_0 or T_{ab} fact. Hence, the proposition is true in this case.

Rule O_2: The set of available messages actually decreases if the rule O_2 is used. Hence if $PCS(k_r, pd, k_0, k_t)$ is not in the view of the intruder at S, it is also not in the view of the intruder at S'.

We have $O_2(pd, -) \notin S$. An inspection of the rule O_2 tells that in order to have $O_2(pd, -) \in S'$, S must contain $N(PCS(k_r, pd, k_0, k_t))$. In that case the view of the intruder at S must contain $PCS(k_r, pd, k_0, k_t)$. Hence if $PCS(k_r, pd, k_0, k_t)$ is not in the view of the intruder at S, $O_2(pd, -) \notin S'$. The transition rule O_2 does not create any O_i facts for all $i \in \{3, com, res?, res1, res2\}$, and does not consume any T_0 or T_{ab} facts. Hence, the proposition is true in this case. $\square$

Proposition 3.4.25. *Let S, S' be reachable states such that S' is obtained from S*

by the application of the role generation rule RG. If me_2, i.e., *$PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S, then me_2,* i.e., *$PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S'.*

Proof. Once again, since R is honest the private key of R is not in the view of the intruder at S, S'. By proposition 3.4.1 $PCS(k_r, pd, k_o, k_t)$ is in the view of the intruder at S, S' only if it is in $analz(A(S))$, $analz(A(S'))$ respectively. If RG is used, then the set of available messages increases by a new constant n' of the sort $unique\_identifier$. Since $analz(n') = \{n'\}$, we get $PCS(k_r, pd, k_o, k_t)$ is in the view of the intruder at S' only if it is in the view of the intruder at S. □

Proposition 3.4.26. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* **I**. *If me_2,* i.e., *$PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S, then me_2,* i.e., *$PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S'.*

Proof. Once again, since R is honest the private key of R is not in the view of the intruder at S, S'. By proposition 3.4.1 $PCS(k_r, pd, k_o, k_t)$ is in the view of the intruder at S, S' only if it is in $analz(A(S))$, $analz(A(S'))$ respectively. The view of the intruder does not change if any rule other than GEN is used (proposition 3.1.2). If the rule GEN is used, then the set of available messages increases by a new constant u of the sort $mssg$. Since $analz(u) = \{u\}$, we get $PCS(k_r, pd, k_o, k_t)$ is in the view of the intruder at S' only if it is in the view of the intruder at S. □

Proposition 3.4.27. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* $\mathbf{R}$*. If R's PCS on pd,* i.e.*, $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S and $R_0(pd), R_{quit}(pd) \in S'$, then $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S'*

Proof. Intuitively, since the intruder does not learn the private key of R, the only way intruder can get hold of R's PCS on pd is when R sends it.

As the key k_{rs} is not in the view of the intruder at S, S' (proposition 3.1.3), $PCS(k_r, pd, k_o, k_t)$ is in the view of the intruder in S, S' only if it is contained in $analz(A(S)), analz(A(S'))$ respectively (proposition 3.4.1).

The proof is again by cases on the rule being used in the transition from S to S'. The rules R_{quit}, R_{ab}, R_{res} R_1, R_3 decreases the number of available messages, and hence $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S' if it is not in the view of the intruder at S.

If the rule R_2 is used to transit from S to S', then the only new available message at S' is $PCS(k_2, pd', k_1, k_3)$ where $pd' =< m', n', k_1, k_2, k_3 >$ where k_1, k_2, k_3 are terms of sort *public_key*, m' is a term of the sort *preagreed_text* and n' is a term of the sort $unique\_identifier$. In this case, $R_2(pd', PCS(k_2, pd', k_1, k_3))$ is contained in S'.

Now, $analz(PCS(k_2, pd', k_1, k_3)) = \{PCS(k_2, pd', k_1, k_3)\} \cup analz(pd')$. Clearly, $PCS(k_r, pd, k_o, k_t)$ is in $analz(A(S'))$ only if pd' is the same as pd. In this case $R_2(pd, -) \in S'$ and by proposition 3.2.21, $R_0(pd)$ and $R_{quit}(pd)$ are not in S'.

Hence, in this case, if $R_0(pd)$ and $R_{quit}(pd)$ are not in S', then $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder in S'.

We can treat R_3 and R_{com} similarly and show that $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder in S' if it is not in the view of the intruder at S. $\square$

Proposition 3.4.28. *For all states S reachable from S_1, if S contains $R_i(pd, -)$ for $i \in \{0, quit\}$, then*

1. *me_2,* i.e., *$PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S,*
2. *$O_i(pd, -) \notin S$ for all $i \in \{2, 3, com, res?, res1, res2\}$, and*
3. *$T_0(pd)$ or $T_{ab}(pd, -) \in S$.*

Proof. Intuitively, R's private key is not known to the intruder. Hence, the intruder cannot create $PCS(k_r, pd, k_o, k_t)$. Since $PCS(k_r, pd, k_o, k_t)$ is never sent, neither T nor O sees it.

We prove this by induction on number of steps that it takes to go from S_1 to S.

Base case: S is S_1. Since the identifier n is freshly generated in the transition from S_0 to S_1, we get, $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S_0. By proposition 3.4.25, $PCS(k_r, pd, k_o, k_t)$ is also not in the view of the intruder at S_1. Since RG does not create an O_i fact for $i \in \{2, 3, com, res?, res1, res2\}$, we get $O_i(pd, -) \notin S$ for all $i \in \{2, 3, com, res?, res1, res2\}$. We also have $T_0(pd) \in S_0$.

Induction Hypothesis: Let the statement of the proposition be true for all states S reachable from S_1 by l steps. Consider a state S' such that S' reachable from S_1

in $l+1$ steps and $R_i(pd,-)$ in S for $i \in \{0, quit\}$. Since S' is reachable from S_1 in $l+1$ steps, there must be a state S such that:

1) S' is reachable from S by a single transition,

2) S is reachable from S_1 by l steps, and

3) by proposition 3.4.21, $R_i(pd,-)$ in S for $i \in \{0, quit\}$.

By induction hypothesis,

a) me_2, *i.e.*, $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S,

b) $O_i(pd,-) \notin S$ for all $i \in \{2, 3, com, res?, res1, res2\}$, and

c) $T_0(pd)$ or $T_{ab}(pd,-) \in S$.

Since $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder at S, then by proposition 3.4.24, the statement of the proposition is true if the transition rule t is in $\mathbf{O} \cup \mathbf{T}$.

By propositions 3.4.27, 3.4.26 and 3.4.25, $PCS(k_r, pd, k_o, k_t)$ is not in the view of the intruder in S' if t is in $\mathbf{R} \cup \mathbf{I} \cup \{RG\}$. Now any transition in $\mathbf{R} \cup \mathbf{I} \cup \{RG\}$ does not create any O_i facts nor consume any T_i facts. Hence, the proposition is true. □

We are ready to prove lemma 3.2.30.

Lemma 3.2.30. *For all states S reachable from S_1,*

1. *If S contains $R_i(pd,-)$ where $i \in \{0, quit\}$, then neither $PCS(k_r, pd, k_o, k_t)$, R's PCS on pd, nor a resolution for pd is in the view of the intruder. If S*

contains $R_i(pd, -)$ where $i \in \{0, quit\}$, then neither me_2 nor a resolution is in the view of the intruder.

2. *If a resolution for pd is in the view of the intruder in S, then for all states S' reachable from S, R has not quit pd in S'.*

Proof. 1. By proposition 3.4.28, if S contains $R_i(pd, -)$, then $PCS(k_r, pd, k_o, k_t)$, is not in the view of the intruder. Also S contains $T_0(pd)$ or $T_{ab}(pd, -)$. By database consistency (lemma 3.2.4), T does not have a resolution for *pd*. Hence by lemma 3.2.5, the resolution is not in the view of the intruder at S.

   If R has quit for *pd* in S, then by proposition 3.4.21, R has quit for *pd* in all states S' reachable from S. Hence, a resolution for *pd* is not in the view of the intruder in all states S' reachable from S.

2. If a resolution for *pd* is in the view of the intruder in S, then for any state S' reachable from S, T has a resolution for *pd* in S' by database persistence (lemma 3.2.3). By database consistency, neither $T_0(pd)$ nor $T_{ab}(pd, -)$ is in S'. By proposition 3.4.28, R has not quit *pd* in S'

□

Proposition 3.4.29. *Let S, S' be reachable states such that*

1) S' is obtained from S by a single transition rule.

2) R does not have an abort_token for pd in S.

3) R has an abort_token for pd in S.

Then an abort_token is on the R-T channel in S, i.e., *S contains $R_n(k_r, k_t, ab\_tok)$.*

Proof. We have R does not have an abort token for *pd* in S, *i.e.*, S does not contain $R_{ab}(pd, -)$. However in S, R does have an abort_token for *pd* in S. Hence R_{ab} is used for the transition from S to S'. An inspection of the rule yields that in order for this rules to apply in S, S must contain $R_n(k_r, k_t, ab\_tok)$. □

Now, we are ready to show lemma 3.2.31.

Lemma 3.2.31. *For all states S reachable from S_1,*

1. *If R has an abort_token for pd then T has an abort_token for pd. Furthermore, in all states S' reachable from S, a resolution for pd is not in the view of the intruder in S'.*

2. *If a resolution for pd is in the view of the intruder in S then for all states S' reachable from S, R does not have an abort_token for pd.*

Proof. 1. Note that in S_1, R does not have an abort_token for *pd*. Proposition 3.4.29 tells us that the only way R gets an abort_token for *pd* is when R reads it from the *R-T* channel. If an abort_token is on the *R-T* channel then it is in the view of the intruder. By lemma 3.2.5, T must have an abort_token for *pd*.

Using this, an easy induction shows that as a result of database persistence, if R has an abort_token for *pd*, then T must have an abort_token for *pd*.

By database persistence and consistency (lemmas 3.2.3, and 3.2.4), in all states reachable from S, T does not have a resolution for *pd*.

2. By lemma 3.2.5, if a resolution is in the view of the intruder, then T must have a resolution for *pd* in S. By database persistence and consistency, (lemmas 3.2.3, and 3.2.4), in all states reachable from S, T does not an abort_token for *pd*. Hence by 1 if S' is a state reachable from S, R does not have an abort_token for *pd* in S.

$\square$

Proposition 3.4.30. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in $\mathbf{O} \cup \mathbf{T}$. If R's signature on pd,* i.e., *$sig(k_r, pd)$, is not in the view of the intruder at S, then R's signature on pd is not in the view of the intruder at S' also.*

Proof. Intuitively, since the intruder does not learn the private key of R, the only way intruder can get hold of R' signature on *pd* is when R sends it.

By proposition 3.1.3, the private key k_{rs} is not in the view of the intruder at S, S'. Hence if $A(S)$ and $A(S')$ are the sets of available messages at S and S' respectively, by proposition 3.4.1, R's signature on *pd* is in the view of the intruder at S, S' only if it is in $analz(A(S)), analz(A(S'))$ respectively. If R's signature on *pd* is not in the view of the intruder at S, it is not in $analz(A(S))$. We shall show by cases on the rule applied to obtain S' that if the transition rule to obtain S' from S is in $\mathbf{O}$, then R's signature on *pd* is not in the $analz(A(S))$. The case when the transition is in **T** can be similarly proved.

1. Rule O_1: A message $PCS(k_1, pd', k_1, k_3)$ is deposited on the network, where k_1, k_2, k_3 are keys of the sort *public_key*, and pd' is $\langle m', n', k_1, k_2, k_3 \rangle$ where m' and n' are terms of the sort *preagreed_text* and *unique_identifier* respectively. (Note pd' may not be the same as pd: there may be several concurrent executions of the protocol). By proposition 3.4.1, $analz(A(S')) = analz(A(S)) \cup analz(PCS(k_1, pd', k_2, k_3))$. Now, $analz(PCS(k_1, pd', k_2, k_3))$ $= \{PCS(k_1, pd', k_2, k_3)\} \cup analz(pd')$. We have $analz(S)$ does not contain R's signature on pd. Clearly, $analz(PCS(k_1, pd', k_2, k_3))$ also does not contain R's signature on pd and hence R's signature on pd is not in $analz(A(S'))$.

2. Rule O_3: A message $sig(k_1, < m', n', k_1, k_2, k_3 >)$ is deposited on the network. R's signature on pd is contained in $analz(S')$ only if $analz(sig(k_2, < m', n', k_1, k_2, k_3 >))$ contains $sig(k_r, pd)$. Clearly, this will be true only if k_1 is same as k_r and $pd =< m', n', k_1, k_2, k_3 >$. But pd is $< m, n, k_o, k_r, k_t >$. Hence k_1 must be k_o. This would mean that k_r and k_o are same. A contradiction since k_r and k_o are different.

3. Rule $O_{res?}$: A message $< PCS(k_1, pd', k_2, k_3), PCS(k_2, pd', k_1, k_3) >$ is deposited on the channel between k_2 and k_3. As in case 1, once again an R's signature on pd is not in the view of the intruder at S'.

4. Rule $O_{ab?}$: A message $sig(k_1, < pd', abort >)$ is deposited on the network,

where k_1, k_2, k_3 are keys of the sort *public_key* and $pd' = \langle m', n', k_1, k_2, k_3 \rangle$, where m' and n' are terms of the sort *preagreed_text* and *unique_identifier* respectively. R's signature on pd is contained in $analz(S')$ only if $analz(sig($ $k_1, \langle pd', abort \rangle))$ contains $sig(k_r, pd)$. Now, $analz(sig(k_1, \langle pd', abort \rangle)) =$ $\{sig(k_1, \langle pd', abort \rangle), \langle pd', abort \rangle, pd', abort, m', n', k_1, k_2, k_3\}$. It can be easily seen that $analz(sig(k_1, \langle pd', abort \rangle))$ does not contain $sig(k_r, pd)$.

5. Rule O_2, O_{com}, O_{ab1}, O_{ab2}, O_{res1}, O_{res2}: The set of available messages actually decreases, hence the view of the intruder at S' does not contain $sig(k_r, pd)$.

$\square$

Proposition 3.4.31. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* **I**. *If R's signature on pd,* i.e., *$sig(k_r, pd)$, is not in the view of the intruder at S, then R's signature on pd is not in the view of the intruder at S' also.*

Proof. Once again, since R is honest the private key of R is not in the view of the intruder at S, S'. By proposition 3.4.1 $PCS(k_r, pd, k_o, k_t)$ is in the view of the intruder at S, S' only if it is in $analz(A(S))$, $analz(A(S'))$ respectively. The view of the intruder does not change if any rule other than GEN is used (proposition 3.1.2).

By proposition 3.1.2, view of the intruder does not change by an application of a rule in **I** other than GEN. Now, if GEN is used, then the set of available messages increases by a new constant u of the sort $mssg$. Since $analz(u) = \{u\}$, we get $sig(k_r, pd)$ is in the view of the intruder at S' only if it is in the view of the intruder at S. $\square$

Proposition 3.4.32. *Let S, S' be reachable states such that S' is obtained from S by the application of the role generation rule RG.*

Proof. Once again, since R is honest the private key of R is not in the view of the intruder at S, S'. By proposition 3.4.1 $PCS(k_r, pd, k_o, k_t)$ is in the view of the intruder at S, S' only if it is in $analz(A(S))$, $analz(A(S'))$ respectively. If RG is used, then the set of available messages increases by a new constant n' of the sort $unique\_identifier$. Since $analz(n') = \{n'\}$, we get $sig(k_r, pd)$ is in the view of the intruder at S' only if it is in the view of the intruder at S. $\square$

Proposition 3.4.33. *Let S, S' be reachable states such that S' is obtained from S by the application of a transition rule in* **R**. *If R's signature on pd,* i.e., *$sig(k_r, pd)$ is not in the view of the intruder at S, then R's signature on pd is in the view of the intruder at S' only if $R_{com}(pd, -) \in S'$.*

Proof. Intuitively, since the intruder does not learn the private key of R, the only way intruder can get hold of R's signature on *pd* is when R sends it.

The proof is again by cases on the rule being used in the transition from S to S'. As in the proof of proposition 3.4.30, we can show that if the transition is anything other than R_{com}, then R's signature on *pd* is not in the view of S' if it is not in the view of the intruder in S.

As the key k_{rs} is not in the view of the intruder at S, S' (proposition 3.1.3), R's signature is in the view of the intruder in S, S' only if it is contained in $analz(A(S))$, $analz(A(S'))$ respectively.

If the rule R_{com} is used to transit from S to S', then the only new available message at S' is $sig(k_2, pd')$ where $pd' =< m', n', k_1, k_2, k_3 >$ where k_1, k_2, k_3 are terms of sort *public_key*, m' is a term of the sort *preagreed_text* and n' is a term of the sort $unique\_identifier$. In this case, $R_{com}(pd', sig(k_2, pd')$ is contained in S'.

Now it can be easily shown that $analz(sig(k_2, pd'))$ is the set that consists of: $sig(k_2, pd'), pd', k_1, k_2, m', n'$ and k_3. Clearly, R's signature for pd is in $analz(A(S'))$ only if it is in $analz(S)$ or pd' is the same as pd. By hypothesis, R's signature on pd is not in the view of the intruder at S. Hence if R's signature on pd is in the view of the intruder at S', then pd' is the same as pd and we get $R_{com}(pd, -) \in S'$. $\square$

Now, we are ready to show lemma 3.2.32.

Lemma 3.2.32. *For all states S reachable from S_1,*

1. *If S contains $R_i(pd, -)$ where $i \in \{0, quit, 1, 2, 3, ab?, ab, res\}$, then R's signature on pd, $sig(k_r, pd)$, is in not in the view of the intruder.*

2. *If R has quit pd or R has an abort_token for pd then for all states S' reachable from S, $sig(k_r, pd)$ in not in the view of the intruder. If $sig(k_r, pd)$ is in the view of the intruder then for all states S' reachable from S, R has neither quit pd nor has an abort_token for pd in S'.*

Proof. 1. Intuitively, by propositions 3.4.30, 3.4.33, 3.4.31 and 3.4.32 if a state transition produces R's signature on pd that was not present before, then R must have sent it.

We prove this by induction on number of steps that it takes to go from S_1 to S.

Base case: S is S_1. Since the identifier n is freshly generated in the transition from S_0 to S_1, clearly R's signature on pd is not in the view of the intruder at S_0. Hence by proposition 3.4.32, it is also not in the view of the intruder at S_1.

Induction Hypothesis: Let the statement of the lemma be true for all states reachable from S_1 by l steps. Consider a state S' such that S' is reachable from S_1 in $l+1$ steps and $R_i(pd, -)$ in S for $i \in \{0, quit, 1, 2, 3, ab?, ab, res\}$. Since S' is reachable from S_1 in $l+1$ steps, there must be a state S such that:

1) S' is reachable from S by a single transition,

2) S is reachable from S_1 by l steps, and

3) By proposition 3.4.21, $R_i(pd, -)$ in S for $i \in \{0, quit, 1, 2, 3, ab?, ab, res\}$.

By induction hypothesis, R's signature on pd is not in the view of the intruder in S. So, if it is in the view of the intruder in S' , then by propositions 3.4.30, 3.4.33, 3.4.31 and 3.4.32, the transition from S to S' must use a rule in **R**. In this case, by proposition 3.4.33, $R_{com}(pd, -) \in S'$. By proposition 3.2.21, S' cannot contain $R_i(pd, -)$ for $i \in \{0, quit, 1, 2, 3, ab?, ab, res\}$.

A contradiction and hence an R's signature on pd is not in the view of the intruder in S' also.

2. If R has quit pd then $R_{quit}(pd, -) \in S$. By proposition 3.4.21, for all states S' reachable from S, $R_{quit}(pd, -) \in S$. By proposition 3.2.21 $R_{com}(pd, -) \notin S'$. Hence, by part 1, $sig(k_r, pd)$ is not in the view of the intruder in S'. If R has an abort_token for pd, then $R_{ab}(pd, -)$. By proposition 3.4.21, for all states S' reachable from S, $R_{com}(pd, -) \in S$. By proposition 3.2.21 $R_{com}(pd, -) \notin S'$. Hence, by part 1, $sig(k_r, pd)$ is not in the view of the intruder in S'.

   If $sig(k_r, pd)$ is in the view of the intruder in S, then by part 1, $R_{com}(pd, -) \in S$. By proposition 3.4.21, for all states S' reachable from S, $R_{com}(pd, -) \in S'$.

□

3.4.7 Recursive characterization of balance

Let A be the honest signer and B be the dishonest signer. Now, we shall show lemmas 3.3.6 and proposition 3.3.7. We need the following proposition.

Proposition 3.4.34. *Let ctr be the continuation tree from S. Given a node N in ctr, let ctr_N be the subtree of ctr rooted at N. We have*

1. *If N is an abort-power node, then there is a selection of removable edges, E_N, in ctr_N such that in each leaf node of $ctr_N \backslash E_N$, A does not have B's signature.*

2. *If N is a resolve-power node, then there is a selection of removable edges, E_N,*

in ctr_N such that in each leaf node of $ctr_N \backslash E_N$, B has A's signature.

Proof. We shall prove part 1. Part 2 can be similarly proved.

Let the height of *ctr* be l. We prove part 1 by backward induction on the height of the node N in ctr.

Base case: Height of node N is l. In this case N is a leaf node in *ctr* and ctr_N is N.

Since N is a leaf node and an abort-power node, we see by definition of an abort-power node that A does not have B's signature in N. Choose E_N to be empty and we get $ctr_N \backslash E_N$ is N. The claim is trivially true in this case.

Induction hypothesis: Suppose the claim is true for each node N' in *ctr* such that

a) N' is an abort-power node, and

b) the height of N' in *ctr* is $\geq i$ for some $0 < i \leq l$.

Let N be a node in *ctr*, such that height of N in *ctr* is $i - 1$. Given a selection, X, of children of N that are connected to N by a removable edge, N_X is the set of children of N that are either in X or connected to N by a non-removable edge. Since N is an abort-power node, by definition there is an selection X such that

either N_X is nonempty and each node in N_X is an abort-power node,

or N_X is empty and A does not have B's signature on *pd* in the state labeling N.

Fix X and consider the two cases separately.

Case a) N_X is nonempty. If $N_1, \ldots, N_j$ are the nodes in N_X, then let $ctr_i, \ldots ctr_j$

be the subtrees of *ctr* rooted at $N_1, \ldots, N_j$ respectively. By induction hypothesis, there are selection of edges $E_1, \ldots, E_j$ such that in $ctr_i \backslash E_1, \ldots ctr_j \backslash E_j$, every leaf node is labeled by a state in which A does not have B's signature.

Now, let X' be the set of children other than X that are connected to N by removable edges. Let E_0 be the set of edges connecting X' to N. Now define E_N be $E_0 \cup E_1 \cup \ldots \cup E_j$. It can be easily seen that $ctr_N \backslash E_N$ is the tree, in which $ctr_i \backslash E_1, \ldots ctr_j \backslash E_j$ are attached to N. There are no other edges coming out of N in $ctr_N \backslash E_N$. Hence, in each leaf node of $ctr_N \backslash E_N$, A does not have B's signature.

Case b) N_X is empty, and A does not have B's signature in N. We let E_N to be the set of all removable edges coming out of N. Then, it can be easily seen that since N_X is empty, we get $ctr_N \backslash E_N$ is the node N. Since, A does not have B's signature in N, the claim is true in this case.

Hence, by induction the claim is true. □

Lemma 3.3.6. *Let ctr be the continuation tree from S. At S, B has the power to abort if and only if the root of ctr is an abort-power node. Also at S, B has the power to complete if and only if the root of ctr is a resolve-power node.*

Proof. We shall show the proof in the case B has power to abort. The other case can be treated similarly.

($\Rightarrow$) Let B have the power to abort at S. Let the height of *ctr* be l.

If E is a selection of removable edges, then $ctr \backslash E$ is the tree obtained from *ctr* by removing E along with all its descendants. B has the power to abort at S. By definition, there is a selection E of removable edges such that in each leaf node of $ctr \backslash E$, A does not have B's signature.

We shall show that each node N in $ctr \backslash E$ is an abort-power node. Since the root is in $ctr \backslash E$, the root will be an abort-power node. We show this by backward induction on the height of the node N in ctr.

Base case: Height is l. In this case N is a leaf node in ctr and hence in $ctr \backslash E$. Therefore if S' labels the node N, by definition A does not have B's signature in S'. N is easily seen to an abort-power node (choose X to be empty in the definition).

Induction hypothesis: Suppose that each node N' in $ctr \backslash E$ is an abort-power node if its height in ctr is $\geq i$ for $0 < i \leq l$.

Now consider a node N in $ctr \backslash E$ such that height of N in ctr is $i - 1$. Now consider the removable edges coming out of N. Given a selection, X, of children of N that are connected to N by a removable edge, N_X is the set of children of N that are either in X or connected to N by a non-removable edge.

Let X be the set of removable edges that are also present in $ctr \backslash E$. We consider two cases: N_X is nonempty and N_X is empty separately.

Case a) N_X is nonempty. Then clearly by construction, each node in N_X must be in $ctr \backslash E$. Also, the height of each node in N_X is i. Hence by induction hypothesis, each node in N_X is an abort-power node and hence by definition N is an abort-power node.

Case b) N_X is empty. Clearly by construction, N must be a leaf node in $ctr \backslash E$.

Hence, A does not have B's signature in the state labeling N, and hence we get by definition N is an abort-power node.

Therefore by induction, each node N in $ctr \backslash E$ is an abort-power node.

($\Leftarrow$) If the root of *ctr* is an abort-power node, then by proposition 3.4.34, there is a selection of edges E such that in each leaf node of $ctr \backslash E$, A does not have B's signature. Hence B has the power to abort at S. $\square$

Now, we are ready to show proposition 3.3.7.

Proposition 3.3.7. *1) If a node N in ctr is labeled by a state in which either A has an abort_token or A has quit, then N is not a resolve-power node.*

2) If a node N in ctr is labeled by a state in which A has B's signature, then N is not an abort-power node.

3) Let N and N' be nodes in ctr such that N' is a child of N and the edge between N and N' is non-removable. If N' is not an abort-power node, then N is not an abort-power node. If N' is not a resolve-power node, then N is not a resolve power node.

Proof. 1) Let N be a node in *ctr* such that in the state labeling N, A has an abort_token for *pd* or A has quit for *pd*. Let ctr_N be the subtree of *ctr* rooted at N. Since the protocol is fair and A is honest, we get that in each node of ctr_N, B does not have A's signature.

If N is a resolve-power node, then by proposition 3.4.34, there is a selection of removable edges E_N in ctr_N such that in each leaf node of $ctr_N \backslash N$, B has A's signature on *pd*. This contradicts the fact that B does not have A's signature in any node in ctr_N. Hence, N is not a resolve-power node.

2) Let N be a node in *ctr* such that in the state labeling N, A has B's signature for *pd*. Let ctr_N be the subtree of *ctr* rooted at N. Since A honest, by propositions 3.4.8 and 3.4.21, we get that in each node of ctr_N, A has B's signature.

If N is an abort-power node, then by proposition 3.4.34, there is a selection of removable edges E_N in ctr_N such that in each leaf node of $ctr_N \backslash N$, A does not have A's signature on *pd*. This contradicts the fact that A has B's signature in every node in ctr_N. Hence, N is not an abort-power node.

3) Clearly, for any selection, X, of removable edges coming out of N, the set N_X will contain N'. If N' is not an abort-power node, by definition, N is not an abort-power node. If N' is not a resolve-power node, by definition, N is not a resolve-power node. □

3.4.8 Balance for honest O

Now, we show lemma 3.3.11 needed to prove balance for honest O. We start by proving a couple of propositions.

Proposition 3.4.35. *Let S be a reachable state, and ctr be the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $O_{res?}(pd, -)$,* i.e., *if O has requested T to resolve and is waiting for a reply, and if T has answered a request on the O-T channel in S', then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

Proof. By proposition 3.2.1 and lemma 3.2.13, either an abort_token for *pd* is on the *O*-*T* Channel or a resolution for *pd* is on the *O*-*T* channel. If the abort_token is on the *O*-*T* channel, then O can read it by the use of rule O_{ab2} and go to a state in which O has an abort_token for *pd*. In this case N has a child N' such that

a) The edge between N and N' is labeled by a rule in **O**. Clearly this edge is non-removable.

b) N' is labeled by a state in which O has an abort_token for *pd*. By proposition 3.3.7, N' is not a resolve-power node.

By proposition 3.3.7, N is not a resolve-power node. Similarly, if a resolution from T is on the *O*-*T* channel, we have N is not an abort-power node. □

Proposition 3.4.36. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $O_{res?}(pd, -)$,* i.e.*if O has requested T to resolve and is waiting for a reply, and if T has yet to answer a request on the O-T channel in S', then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

Proof. By proposition 3.2.1 and lemma 3.2.13, the request for *pd* is on the *O*-*T* Channel. Since T has yet not answered a request on the *O*-*T* channel in S, it can answer it and we obtain a state in which T has answered a request on the *O*-*T* channel. We get N has a child N' such that

a) The edge between N and N' is labeled by a rule in **T**. Clearly this edge is non-removable.

b) N' is labeled a state which contains $O_{res?}(pd-)$ and in which T has answered a request on the O-T channel.

By proposition 3.4.35 either N' is not an abort-power node or N' is not a resolve power node. By proposition 3.3.7, if N' is not an abort-power node, then N is not an abort-power node. If N' is not a resolve-power node, then N is not a resolve-power node. □

We can combine the above two propositions and 3.2.13 to prove lemma 3.3.11.

Lemma 3.3.11. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $O_{res?}(pd, -)$,* i.e., *if O has requested T to resolve and is waiting for a reply, then*

either N is not an abort-power node, or N is not a resolve-power node.

Proof. By lemma 3.2.13, T has either answered a request for *pd* on the O-T channel in S', or T is yet to answer a request for *pd* on the O-T channel in S'. The result now follows by propositions 3.4.35 and 3.4.36. □

3.4.9 Balance for honest R

Now, we show lemma 3.3.11 needed to prove balance for honest R. We start by proving a couple of propositions.

Proposition 3.4.37. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $R_{res?}(pd, -)$,* i.e., *if R has requested T to resolve and is waiting for a reply, and if T has answered a request on the R-T channel in S', then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

Proof. By proposition 3.2.1 and lemma 3.2.27, either an abort_token for *pd* is on the *R-T* Channel or a resolution for *pd* is on the *R-T* channel. If the abort_token is on the *R-T* channel, then R can read it by the use of rule R_{ab2} and go to a state in which R has an abort_token. In this case N has child N' such that

a) The edge between N and N' is labeled by a rule in **R**. Clearly this edge is non-removable.

b) N' is labeled a state in which R has an abort_token for *pd*. By proposition 3.3.7, N' is not a resolve-power node.

By proposition 3.3.7, N is not a resolve-power node. Similarly, if a resolution from T is on the *R-T* channel, we have N is not an abort-power node. □

Proposition 3.4.38. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $R_{res?}(pd, -)$,* i.e., *if R has requested T to resolve and is waiting for a reply, and if T has yet to answer a request on the R-T channel in S', then*

<u>either</u> N is not an abort-power node, <u>or</u> N is not a resolve-power node.

Proof. By lemma 3.2.13, the request for *pd* is on the *R-T* Channel. Since T has yet not answered a request on the *R-T* channel in S, it can answer it and we obtain a state S', in which T has answered a request on the *R-T* channel. We get N has a child N' such that

a) The edge between N and N' is labeled by a rule in $\mathbf{T}$. Clearly this edge is non-removable.

b) N' is labeled by a state that contains $R_{res?}(pd-)$ and in which T has answered a request on the R-T channel.

By proposition 3.4.37 either N' is not an abort-power node or N' is not a resolve power node. By proposition 3.3.7, if N' is not an abort-power node, then N is not an abort-power node. If N' is not a resolve-power node, then N is not a resolve-power node. □

We can combine the above two propositions and 3.2.27 to prove lemma 3.3.14.

Lemma 3.3.14. *Let S be a reachable state, and ctr the continuation tree from S. Let N be a node in ctr, labeled by a state S'. If S' contains $R_{res?}(pd,-)$,* i.e.*, if R has requested T to resolve and is waiting for a reply, then*

$\underline{\text{either}}$ N is not an abort-power node, $\underline{\text{or}}$ N is not a resolve-power node.

Proof. By lemma 3.2.27, T has either answered a request for *pd* on the R-T channel in S', or T is yet to answer a request for *pd* on the R-T channel in S'. The result now follows by propositions 3.4.37 and 3.4.38. □

Chapter 4

Model for general signature-exchange protocols

A detailed study of the two-party signature protocol in chapter 2 shows that the protocol is fair, effective and balanced for honest signers (see chapter 3). Hence, the protocol in chapter 2 does not provide any dishonest signer the ability to control the outcome of the protocol if the other signer is honest. A key ingredient in the proof of balance is that an honest signer may non-deterministically exercise any of the options available to it, including contacting the trusted third party for error recovery.

As we argue in chapter 1, this does not cover all the nuisances of the natural behavior of an honest signer. For example, an honest signer may trust its counter-party and prefer to wait for their messages instead of rushing to the adjudicating party. Indeed, as we argue in chapter 1, the value of an optimistic protocol lies in the frequency with which "optimistic" signers can complete the protocol without using the third party.

In this chapter, we give a refined game-theoretic model, that can be used to reason about signer bias and reason about protocol properties (see chapter 5). In this refined framework, we add signals that the signers use to decide when to contact the trusted third party. Another departure from the analysis of the protocol in chapter 2 is the way we model dishonest behavior. Instead of having the dishonest signer share its key with the intruder, we equip each of the signers with additional dishonest moves. Finally, we limit our attention to single protocol runs. Since our impossibility result (see chapter 6) holds even for a single run, it shall also hold for concurrent runs.

The protocol formalism is once again multiset rewriting with existential quantification, MSR [14, 13, 22]. MSR rules shall model protocol actions and states will model the world. In an interleaving semantics of concurrency, we can commute the order of application of transition rules that affect independent parts of the system:

Proposition 4.0.39. *Let S_1 and S_2 be disjoint multisets and let S be their union,* i.e.*, let $S = S_1 \uplus S_2$. If,*

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. The proof follows immediately from the definition of the multiset-rewriting formalism. □

We start by describing some basic sorts that are used in our refined model.

Keys, messages, channels. We assume that our vocabulary contains two sorts: *public_key* and *mssg*. The sort *public_key* is used for public keys of the protocol participants, and the sort *mssg* is used for the protocol messages. We use $k, k', k_a, k_1, \ldots$ to range over values of the sort *public_key* and $m_1, m_2, \ldots$ to range over values of the sort *mssg*. Participants of a protocol run are identified with their public keys.

Channels between signers are formalized by unary predicates whose argument is of the sort *mssg*. We call such predicates *network predicates.* For the class of protocols we study, we need only one network predicate. We reserve N for this network predicate. Channels to trusted third party are formalized by ternary predicates whose first two arguments are of the sort *public_key* and the third argument is of the sort *mssg*. We call such predicates *TTP_channel predicates* and use $P_n, S_n, \ldots$ to range over such predicates. For example, $P_n(k_1, k_2, m_1)$ indicates a channel between a signer identified with public key k_1, and a trusted third party identified with public key k_2, carrying a message m_1. (Please note that since the arity of the network predicate N and TTP_channel predicates are different, TTP_channel predicates are distinct from N.)

Pre-agreed texts, unique identifiers We assume that our vocabulary also contains further two sorts: $Preagreed\_text$ and $unique\_identifier$ as sub-sorts of the sort $mssg$. The sort *preagreed_text* is used for the pre-agreed texts and $m, m', \ldots$ are used to range over values of the sort *preagreed_text*. We assume that the signers use a globally unique identifier for each protocol instance and the sort $unique\_identifier$ is used for these identifiers. We use $n, n', \ldots$ to range over values of the sort $unique\_identifier$.

We assume that our vocabulary also contains a sort: *protocol_instance* and a function: $< \_, \_, \_, \_, \_ >$: *public_key* $\times$ *public_key* $\times$ *public_key* $\times$ *preagreed_text* $\times$ $unique\_identifier \rightarrow$ *protocol_instance*. The sort *protocol_instance* identifies the signers, the trusted third party, pre-agreed text and the globally unique identifier in a protocol instance. We use $pd, pd', \ldots$ to range over values of the sort *protocol_instance*. For example, the protocol instance $pd =< k_o, k_r, k_t, m, n >$ describes the protocol instance in which signers with public keys k_o and k_r, are trying to exchange signatures on the pre-agreed text, m, using the globally unique identifier, n, with the help of trusted third party identified with public key k_t.

Signals. We assume that our vocabulary also contains further a sort: *timer_state*. We assume that the signature has exactly three constants: *unset*, *set*, and *timed_out* of the sort *timer_state*. We use $ts, ts', \ldots$ to range over constants of the sort *timer_state*.

Timers are formalized by binary predicates whose first argument is of the sort *public_key*, the second argument is of the sort *timer_state*. We call such predicates *timer predicates* and use $Z, Z_1, Z_2, \ldots$, to range over timer predicates. For example, the fact $Z(k, ts)$ indicates a timer, Z, belonging to the participant identified with public key k in the timer state ts.

In our formalism timers should be thought of as local signals. They do not refer to any global synchronous time. These signals are used by signers to decide when to quit waiting for a message from the opponent. These signals help in modeling signer bias (see section 5.4).

Cryptographic primitives. A number of cryptographic primitives may be used in signature-exchange protocols, including encryption, signatures, hash functions, and more specialized functions such as the PCS signature function that we encountered in chapter 2. The purpose of cryptography, in general, is to provide messages that are meaningful to some parties, but not subject to arbitrary (non-polynomial-time) computation by any parties. For example, encryption provides messages that are meaningful to any recipient with the decryption key, but not subject to decryption by any agent that 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.

For each operation used in messages of a protocol, we assume there is some MSR characterization of the computability properties of this operation. To give a concrete framework for presenting these rules, let us assume some set of predicates $\mathcal{P} = \{\mathcal{P}^\alpha | \alpha \text{ any sort}\}$. Since the sort α is determined by the sort of the arguments to P^α, we will not write the sort when it is either irrelevant or clear from context.

Intuitively, a rule of the form

$$P(s_1), \ldots, P(s_m), F_1, \ldots, F_j \longrightarrow P(t_1), \ldots, P(t_n), F_1, \ldots, F_j$$

means that if an agent possesses data $s_1, \ldots, s_m$, then under conditions specified by facts $F_1, \ldots, F_j$, it is computationally feasible for the agent to possess $t_1, \ldots, t_n$. Some example rules of possession are the familiar "Dolev-Yao" rules given in [14, 13, 22]:

$$\begin{aligned} P^{\text{mssg}}(x), P^{\text{key}}(k) &\longrightarrow P^{\text{mssg}}(\text{encrypt}(k, x)) \\ P^{\text{mssg}}(\text{encrypt}(k, x)), P^{\text{key}}(k^{-1}), \text{Keypair}(k, k^{-1}) &\longrightarrow P^{\text{mssg}}(x) \end{aligned}$$

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.

For invertible operations, such as pairing, that may be used in protocol messages, we assume MSR possession rules that state that a pair may be computed from its parts and, conversely, given a pair, its parts may be computed.

In the remainder of the thesis, we assume some fixed theory **Possess** of rules that characterize the computationally feasible operations on messages defined by combinations of functions provided in the vocabulary used to present the protocol. As a disclaimer, we emphasize that the results in this thesis apply to a signature-exchange protocol using cryptographic primitives only to the extent that **Possess** accurately characterizes the computationally feasible operations in those primitives. 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 negative results or lower bounds.

4.1 Modeling protocol participants

A *theory* is a set of MSR rules. We use $\mathbf{A}, \mathbf{B}, \ldots$ to range over theories. A signature exchange protocol has three theories, two for the parties trying to exchange signatures and one for the trusted third party. These theories describe the valid protocol steps for the signers and the trusted third party.

Definition 4.1.1. A theory, $\mathbf{A}$, is said to be a *role theory for participant A identified with public key k_a*, where k_a is a constant of the sort *public_key*, if $\mathbf{A}$ satisfies the following conditions.

i) There is a finite list of predicates, called the *role state predicates* and numbered $A_0, A_1, \ldots, A_n$ for some n, and two finite lists of timer predicates, collectively called the *timers of A*.

ii) The role state predicates, the timer predicates in the first list of timers of A and the timer predicates in the second list of timers of A are all mutually distinct. A_0 is a binary predicate whose first argument is of the sort *public_key* and the second argument is of the sort *protocol_instance*.

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, it must be the case that $i < j$. If A_0 occurs in l, then $A_0(k_o, pd) \in l$, for some term pd of the sort *protocol_instance*.

2. If A_j 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 $A_i(u_1, \ldots, u_k) \in l$ and $A_j(v_1, \ldots, v_m) \in r$, then u_p and v_p are the same terms for all $1 \leq p \leq k$.

3. Let $A_i(u_1, \ldots, u_m) \in l$, $A_j(v_1, \ldots, v_m) \in r$. Let *messages* be the set of terms, u, such that $N(u) \in l$ or $P_n(k_1, k_2, u) \in l$ for some TTP_channel predicate P_n. For each p, v_p is derivable from $u_1, \ldots, u_m$ and *messages* using the rules in

Possess.

4. For each timer of A, say Z, it is the case that l and r each contain at most one occurrence of Z. If Z occurs in r, then it occurs in l also.

5. Each occurrence of a timer of A, say Z, in l or in r is of the form $Z(k_a, ts)$, where ts is a constant of the sort *timer_state*.

6. For each timer of A, say Z,

   i) if $Z(k_a, unset) \in l$, then either $Z(k_a, unset) \in r$, or $Z(k_a, set) \in r$,

   ii) if $Z(k_a, set) \in l$, then $Z(k_a, set) \in r$,

   iii) if $Z(k_a, timed\_out) \in l$, then $Z(k_a, timed\_out) \in r$.

7. If a network predicate or a TTP_channel predicate occurs in l, then r does not contain an occurrence of a network predicate or a TTP_channel.

8. If a network predicate or a TTP_channel predicate occurs in r, then l does not contain an occurrence of a network predicate or a TTP_channel.

9. For any predicate P other than a role state predicate, timer predicate, a network predicate, or a TTP_channel predicate, an atomic formula $P(t_1, ..., t_n)$ has the same occurrences in l as in r.

A_0 is called the *initial role state of A*. Timer predicates in the first list of timers of A are called *abort timers of A* and are numbered $Z_{abort,1}, Z_{abort,2}, \ldots Z_{abort,p}$ for some p. Timer predicates in the second list of timers of A are called *resolve timers* *of* A and are numbered $Z_{resolve,1}, Z_{resolve,2}, \ldots Z_{resolve,m}$ for some m. The transitions mentioned in condition 7 are called *receive transitions* and the transitions mentioned in condition 8 are called *send transitions*. The predicates P mentioned in condition 9 are called *persistent predicates* and the facts in which such predicates occur are called *persistent facts*.

Definition 4.1.2. If Z is a timer of a participant A with public key k_a, then the *time-out rule* of the timer Z is $Z(k_a, set) \rightarrow Z(k_a, timed\_out)$.

Definition 4.1.3. A theory $\mathbf{P}$ is a *protocol theory for signers* O *and* R *with public keys* k_o *and* k_r *respectively, and trusted third party* T *with public key* k_t, where k_o, k_r, k_t are constants of the sort *public_key*, if $\mathbf{P} = \mathbf{O} \uplus \mathbf{R} \uplus \mathbf{T_0} \uplus \mathbf{O_{timeouts}} \uplus \mathbf{R_{timeouts}} \uplus \mathbf{T_{timeouts}}$, where

1. $\mathbf{O}$ is a role theory for participant O with public key k_o, such that there is at most one TTP_channel predicate occurring in $\mathbf{O}$, say P_n. P_n may occur in $\mathbf{T_0}$ but not in $\mathbf{R}$. Furthermore, each occurrence of P_n in $\mathbf{O}$ or in $\mathbf{T_0}$ is of the form form $P_n(k_o, k_t, m_1)$, where m_1 is a term of the sort *mssg*.

2. $\mathbf{R}$ is a role theory for participant R with public key k_r, such that there is at most one TTP_channel predicate occurring in $\mathbf{R}$, say S_n. S_n may occur in $\mathbf{T_0}$ but not in $\mathbf{O}$. Furthermore, each occurrence of S_n in $\mathbf{R}$ or in $\mathbf{T_0}$ is of the form form $S_n(k_r, k_t, m_1)$, where m_1 is a term of the sort *mssg*.

3. $\mathbf{T_0}$ is a role theory for participant T with public key k_t, such that if a

TTP_channel predicates occurs in $\mathbf{T_0}$, then it occurs in $\mathbf{O}$ or in $\mathbf{R}$.

4. The role state predicates and the timers of O do not occur in $\mathbf{R}$ and $\mathbf{T_0}$.

5. The role state predicates and the timers of R do not occur in $\mathbf{O}$ and $\mathbf{T_0}$.

6. The role state predicates and the timers of T do not occur in $\mathbf{O}$ and $\mathbf{R}$.

7. $\mathbf{O_{timeouts}}$ is the set of time-out rules of all timers of O,

8. $\mathbf{R_{timeouts}}$ is the set of time-out rules of all timers of R,

9. $\mathbf{T_{timeouts}}$ be the set of time-out rules of all timers of T.

For the rest of this chapter we pick a protocol theory, say $\mathbf{P}$, and fix it. Let $\mathbf{P} = \mathbf{O} \uplus \mathbf{R} \uplus \mathbf{T_0} \uplus \mathbf{O_{timeouts}} \uplus \mathbf{R_{timeouts}} \uplus \mathbf{T_{timeouts}}$ be the protocol theory for signers O and R with public keys k_o and k_r and trusted third party T with public key k_t. $\mathbf{O}$, $\mathbf{R}$, and $\mathbf{T}$ are role theories for O, R, and T with public keys k_o, k_r and k_t respectively. Let $\mathbf{T} = \mathbf{T_0} \cup \mathbf{T_{timeouts}}$. For the rest of the thesis, we shall be using only $\mathbf{T}$ and not refer to the role theory and timeouts of T separately.

4.2 Threat model

We are interested in security guarantees provided by signature- exchange protocols when one of the signers misbehaves in an arbitrary way. For the purposes of this paper, we assume that the channels between signers and the trusted third party are protected cryptographically according to the protocol specification [3, 28] (in contrast to [35] that considers security of protocols under various relaxations of channel quality). We assume that the adversary may intercept messages from the network, decompose these messages and construct new messages. Furthermore, we assume that the adversary may stop prematurely and can ignore the state of the timer predicates. We equip the dishonest party with *dishonest moves* in addition to the protocol theory. $\mathbf{O_{dishonest}}$ is the set of dishonest rules of O, and $\mathbf{R_{dishonest}}$ is the set of dishonest rules of R. The basic form of these theories is described in appendix 4.2. T is assumed to be honest.

Although the precise formulation of the dishonest moves depend on the message format and the cryptographic primitives used in the protocol specification, we shall describe here the basic form of the dishonest moves. In particular, we describe the basic form of the theory $\mathbf{O_{dishonest}}$, The dishonest rules for R can be similarly described.

In order to model the dishonest rules, we assume that there is a binary predicates M whose first argument is of the sort *public_key* and the second argument is of the sort *mssg*. M represents additional memory of a dishonest signer.

A dishonest O would process data in two phases: first phase is to gather data and the next phase is to perform cryptographic operations and send data. Recall that $\mathbf{O}$ has contains a network N and a trusted third party channel P_n. A dishonest O can gather data from the network, N, or from the trusted third party channel P_n, or from any role state O_i. Let x, x' be variables of the sort *mssg*. If O_i is a *k-ary* predicate, whose arguments are of the sort $s_1, s_2, \ldots, s_k$, then pick k variables $x_1, x_2, \ldots, x_k$ of the sort $s_1, s_2, \ldots, s_k$ respectively. The rules of gathering data are:

$$\begin{array}{lcl} N(x) & \longrightarrow & M(k_o, x) \\ P_n(k_o, k_t, x) & \longrightarrow & P_n(k_o, k_t, x), M(k_o, x) \\ O_i(x_1, x_2, \ldots, x_k) & \longrightarrow & O_i(x_1, x_2, \ldots, x_k), M(k_o, x_1), M(k_o, x_2), \ldots, M(k_o, x_k) \end{array}$$

According to protocol specification, a dishonest O may also be able to bock data on the O-T channel:

$$P_n(k_o, k_t, x) \longrightarrow M(k_o, x)$$

According to protocol specification, a dishonest O may be able to learn, insert or block data on the channel between R and T. Let S_n be the trusted third party channel in **R**. The following rule, for example allows O to gather data on S_n:

$$S_n(k_r, k_t, x) \quad \longrightarrow \quad S_n(k_r, k_t, x), M(k_o, x)$$

The cryptographic operations that a dishonest O may perform will depend on the cryptographic operations being used. In order to give a concrete set of rules that model these, we refer to the fixed theory **Possess**. For each rule in **Possess**, we have a corresponding rule in $\mathbf{O_{threat}}$ which reflects the cryptographic operation. For example, suppose that **Possess** contains the following rule:

$$P(s_1), \ldots, P(s_m), F_1, \ldots, F_j \longrightarrow P(t_1), \ldots, P(t_n), F_1, \ldots, F_j$$

Then if dishonest O has $s_1, \ldots, s_m$ in its memory, then a dishonest O can construct

$t_1, \ldots, t_n$:

$$\begin{aligned} M(k_o, s_1), \ldots, M(k_o, s_m), F_1, \ldots, F_j \longrightarrow \; & M(k_o, s_1), \ldots, M(k_o, s_m), \\ & M(k_o, t_1), \ldots, M(k_o, t_n), F_1, \ldots, F_j \end{aligned}$$

The rules for sending are:

$$\begin{aligned} M(k_o, x) & \longrightarrow M(k_o, x), N(x) \\ M(k_o, x) & \longrightarrow M(k_o, x), P_n(k_o, k_t, x) \end{aligned}$$

Again, whether O can write on the channel between R and T will depend on the protocol specification.

A dishonest O may also create new data:

$$\longrightarrow \exists x M(k_o, x)$$

We shall assume that quitting, that is refusing to take further part in protocol execution is dishonest behavior. To quit from a role state, say O_i, which is a *k-ary* predicate whose arguments are of the sorts $s_1, s_2, \ldots, s_k$ respectively, we introduce in our signature a *k-ary* predicate $O_{dquit,i}$ whose arguments are of the sort $s_1, s_2, \ldots, s_k$. We also introduce k variables $x_1, x_2, \ldots, x_k$ of the sort $s_1, s_2, \ldots, s_k$ respectively, and introduce the following rule in $\mathbf{O_{dishonest}}$:

$$O_i(x_1, x_2, \ldots, x_k) \longrightarrow O_{dquit,i}(x_1, x_2, \ldots, x_k), M(k_o, x_1), M(k_o, x_2), \ldots, M(k_o, x_k)$$

If an honest O may perform an action depending upon the state of a timers, a dishonest O should also be able to disregard the state of some or all of these timers. For example, suppose that the following rule is in $\mathbf{O}$ (here Z is timer predicate):

$$O_i(\vec{u}), V_1(\vec{s_1}), \ldots, V_k(\vec{s_k}), Z(k_o, ts) \longrightarrow O_j(\vec{u}), W_1(\vec{t_1}), \ldots, W_l(\vec{t_l}), Z(k_o, ts')$$

Then, dishonest O may ignore the timer Z:

$$O_i(\vec{u}), V_1(\vec{s_1}), \ldots, V_k(\vec{s_k}) \longrightarrow O_j(\vec{u}), W_1(\vec{t_1}), \ldots, W_l(\vec{t_l})$$

4.3 Initial set of facts, protocol definition, traces

In addition to the role theories, the timeout rules and dishonest rules for the signers, a protocol specification also includes an *initial* <u>*set*</u> *of facts*, say S_0, describing the initial state of the world. S_0 contains the initial role states of O, R, T, and the initial state of the timers.

In particular, the signers have agreed upon the on the text m, and globally unique 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 **O**, **R**, and **T** respectively and pd is the term $< k_o, k_r, k_t, m, n >$. There is no other occurrence of a role state predicate in S_0.

2. For each timer predicate, say Z, of O, R, or T, there is exactly one occurrence of Z in S_0.

3. Either $Z(k_o, unset)$ or $Z(k_o, set)$ but not both, for each timer Z of O.

4. Either $Z(k_r, unset)$ or $Z(k_r, set)$ but not both, for each timer Z of R.

5. Either $Z(k_t, unset)$ or $Z(k_t, set)$ but not both, for each timer Z of T.

6. $M(k_o, m), M(k_o, n), M(k_r, m), M(k_r, n)$.

A *protocol definition* then consists of a set of possession rules **Possess**, a protocol theory $\mathbf{P}$, two theories, $\mathbf{O_{threat}}$ and $\mathbf{R_{threat}}$ that model dishonest behavior of signers, and an initial set of facts S_0. Unlike in chapter 2, we do not have a role generation theory since we are only considering one protocol instance. For the rest of the chapter, we fix our protocol definition.

A protocol *trace* from a state S is a labeled chain such that each node in the chain is labeled by a state, a finite multiset of facts, with the root being labeled by S. The edge connecting any two nodes is labeled by a 3-tuple. The first argument of this tuple is a state transition rule, second argument is a ground substitution and the third argument is a theory in the set $\{\mathbf{O}, \mathbf{R}, \mathbf{T}, \mathbf{O_{timeouts}}, \mathbf{R_{timeouts}}, \mathbf{O_{threat}}, \mathbf{R_{threat}}\}$. Furthermore, it must be the case that if $< t, \sigma, \mathbf{Q} >$ labels the edge connecting nodes labeled by S_1 and S_2, then the application of t using σ transforms S_1 to S_2, and $t \in \mathbf{Q}$. For simplicity sake, we shall sometimes say that the edge is labeled by a transition in $\mathbf{Q}$ if $t \in \mathbf{Q}$. Any state labeling a node in this chain is said to be *reachable from* S. Any state reachable from the initial set of facts S_0 is said to *reachable.*

A *full* trace is a finite trace terminating in a node labeled by a state, say S', in which all timers are in *unset* or *timed-out* state, *i.e.*, if Z is a timer of O or R or T, then $Z(k, set) \notin S'$, for any term k of the sort *public_key*.

An edge in the trace *tr* is said to be a *dishonest move of* O if it is labeled by a transition in $\mathbf{O_{threat}}$. O is said be *honest in a trace* if there are no dishonest moves of O in the trace. If S is reachable from S_0 by a trace in which O is honest, then S is said to be $r$eachable for honest O. When we say that O is honest, we consider only traces in which O is honest and states reachable for honest O. We can define traces in which R is honest and states reachable for honest R similarly.

4.4 Possession

Let A be one of the signers, O or R, S be a reachable state and A_i be the internal role state of A in S. At S, we say A *possess* u if u is derivable from the terms in the internal role state predicate A_i in S, and the dishonest memory of A in S using the rules in **Possess**. Let $Possess(A, S)$ be the set of terms that A possesses at S. We show that possession is always monotonic, and possession increases only if a message is read.

Proposition 4.4.1. *Let S, S' be two reachable states such that S' is obtained from S by using a transition rule t. We have:*

1. *$Possess(A, S) \subseteq Possess(A, S')$.*
2. *If $Possess(A, S) \subsetneq Possess(A, S')$, then $t \in \mathbf{A} \cup \mathbf{A_{threat}}$.*
3. *If $Possess(A, S) \subsetneq Possess(A, S')$, then the left hand side of the transition rule t contains an occurrence of the network predicate or a TTP_channel predicate.*

Proof. Assume that A is honest. (The case where A is dishonest is similar).

1. Intuitively, A accumulates all the data that it has read before.

   t may lie in one of the following: $\mathbf{A}, \mathbf{B}, \mathbf{T}, \mathbf{A_{timeouts}}, \mathbf{B_{timeouts}}, \mathbf{B_{threat}}$

   By definition of possession, A's possession changes in S' from S only if the terms in the internal role state predicate of A changes in S' from S. Since, the internal role state predicates of A do not occur in each of $\mathbf{B}, \mathbf{T}, \mathbf{A_{timeouts}}$, $\mathbf{B_{timeouts}}$, and $\mathbf{B_{threat}}$, if A's possession changes, t must belong to $\mathbf{A}$.

   In the role theory of A, we ensured that whenever the internal role state changes, the terms that are in the internal role state predicate are always carried over to the new internal role state predicate (definition 4.1.1, condition 2). Hence, all the terms that are constructible from terms in the internal role state predicate of A in S are also constructible in S'. Therefore, $Possess(A, S) \subseteq Possess(A, S')$.

2. If the possession of A changes, then we saw in the proof of part 1 that t must belong to $\mathbf{A}$. Hence if $Possess(A, S) \subsetneq Possess(A, S')$, then $t \in \mathbf{A} \cup \mathbf{A_{threat}}$.

3. Intuitively, the only way A constructs new terms is when it receives some data that it cannot construct before.

   Let $u_1, \ldots, u_k$ be the terms that occur in internal role state predicate of A in S, and $v_1, \ldots, v_m$ be the terms that occur in internal role state predicate of A in S'. If the left hand side of t does not contain any occurrence of a network predicate and $TTP\_channel$ predicate, then $v_1, \ldots, v_m$ must be constructible from $u_1, \ldots, u_k$ using rules in **Possess** (definition 4.1.1, condition 3). By definition of possession, $Possess(A, S') \subseteq Possess(A, S)$ and thereby using part 1, $Possess(A, S) = Possess(A, S')$.

Therefore, if $Possess(A, S) \subsetneq Possess(A, S')$, then the left hand side of the rule t contains an occurrence of the network predicate or a TTP_channel predicate.

□

4.5 Strategies

In order to define fairness and advantage, we adapt the notion of games and strategies from classical game theory. Our game consists of three players and strategies involve coalitions amongst them. There is a player each for the signers O and R, and a player for T. A signer may be honest or potentially dishonest. If a signer A is honest then we shall only consider traces in which A is honest.

In addition to its own actions, a signer A may control (in a sense that shall be made more precise shortly) some or all of its own timeouts as well as timeouts of counterparty. This shall help us model player bias formally (see section 5.4). Towards this end, we shall divide the timeout rules of each signer into two sets: one for timeouts of abort timers and one for timeouts of resolve timers. Let $\mathbf{O_{atimeouts}}$ be the set of timeouts of all abort timers of O, $\mathbf{O_{rtimeouts}}$ the set of timeouts of all resolve timers of O, $\mathbf{R_{atimeouts}}$ the set of timeouts of all abort timers of R, and $\mathbf{R_{rtimeouts}}$ the set of timeouts of all resolve timers of R.

As in section 3.3, full continuation trees represent all the possible plays and truncated continuation trees represent strategies of coalitions. Now, we define coalitions and strategies formally.

Assuming that a dishonest signer makes only a finite number of (possibly dishonest) moves, a *continuation tree*, ctr, from S, is the tree of all the possible full traces. If a signer A is honest, then we shall require that S be reachable for honest A, and ctr be the tree of all the traces in which A is honest. This continuation tree can be thought of as a game tree that represents the complete set of possible plays. If E is a subset of edges of ctr, let $ctr \backslash E$ be the tree obtained from ctr by removing the edges in E along with all of their descendants.

In our game tree ctr, some moves are under the control of each signer. Let A be one of the signers, O or R, and B be the other signer. If A is O, then $\mathbf{A}$ is $\mathbf{O}$, otherwise it is $\mathbf{R}$. Similarly if B is O then $\mathbf{B}$ is $\mathbf{O}$, otherwise it is $\mathbf{R}$. Any edge e in ctr that is labeled by a rule in $\mathbf{A}$ is *under A's control.* If A is dishonest then any edge labeled by a rule in $\mathbf{A_{threat}}$ is also *under A's control.* Unless otherwise specified, these are the only edges under A's control.

In order to define optimistic protocols (see section 5.3) and model player bias (see section 5.4), we will require that A control some timeouts. Hence, the following edges may also be *under A's control*:

1. Any edge that is labeled by a transition in $\mathbf{O_{atimeouts}}$. In this case, we say that A controls the timeouts of abort timers of O.

2. Any edge that is labeled by a transition in $\mathbf{O_{rtimeouts}}$. In this case, we say that A controls the timeouts of resolve timers of O.

3. Any edge that is labeled by a transition in $\mathbf{R_{atimeouts}}$. In this case, we say that A controls the timeouts of abort timers of R.

4. Any edge that is labeled by a transition in $\mathbf{R_{rtimeouts}}$. In this case, we say that A controls the timeouts of resolve timers of R.

No other edges are under A's control.

Any edge e in *ctr* that is labeled by a rule in $\mathbf{T}$ is under *T's control.* No other edges are under T's control.

Definition 4.5.1. Let $X \subseteq \{A, B, T\}$, and E be a subset of edges of *ctr* such that each edge in E is in under the control of some $p \in X$. *ctr*$\backslash E$ is said to be a *strategy for the coalition X* if all the paths in *ctr*$\backslash E$ from root to leaf nodes are full traces.

If there are no dishonest moves of any $p \in X$ in *ctr*$\backslash E$, then *ctr*$\backslash E$ is said to be an *honest strategy.*

Please note that the above definition is the standard notion of strategies from game theory. E represents the actions that the coalition X considers unfavorable and *ctr*$\backslash E$ represents the continuations that X prefers. Also note that because of the fullness requirement on traces, all timers must time out eventually. However if a timeout is controlled by some signer in X, when it happens may depend on X's strategy.

For fairness and advantage, the coalition X will be interested in primarily two kinds of goals: one to ensure that a signer gets the counterparty's signature on the pre-agreed text and one in which a signer does not the counterparty's signature.

Definition 4.5.2. A state S is *successful* for A if A possesses B's signature on the pre-agreed text in S, and *unsuccessful* otherwise.

Definition 4.5.3. If $\exists$ a strategy $ctr\backslash E$ for coalition X such that all leaf nodes are labeled by states successful for A, 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.*

Definition 4.5.4. If $\exists$ a strategy $ctr\backslash E$ for A such that all leaf nodes are labeled by states successful for A and B, then A has a *strategy from S to exchange signatures with B.*

Definition 4.5.5. If $\exists$ a strategy $ctr\backslash E$ such that all leaf nodes are labeled by states unsuccessful for honest A, then X *has a strategy from S to prevent A from obtaining B's signature.*

4.6 GJM protocol with timers

Now, we shall describe the protocol theory for protocol in chapter 2, when the signers use timers to decide when to quit waiting for the other signer. For simplicity, we just describe $\mathbf{O}$, the role theory of O and $\mathbf{O_{timeouts}}$ the theory of timeouts of timers of O. The theories, $\mathbf{O_{threat}}, \mathbf{R}, \mathbf{R_{threat}}, \mathbf{T}$, and $\mathbf{R_{timeouts}}$ can be similarly defined.

$\mathbf{O}$ is given in table 4.1. $\mathbf{O}$ has the role state predicates O_0, O_1, O_2, O_3, O_{com}, $O_{ab?}$, $O_{res?}$, O_{ab1}, O_{res1}, O_{ab2}, and O_{res}. $\mathbf{O}$ has a single abort timer predicate Z_{ab} and a single resolve timer predicate Z_{res}. Furthermore, the network predicate N occurs in $\mathbf{O}$ that models the network, and a TTP_channel predicate P_n occurs in $\mathbf{O}$ which models the O-T channel.

The numbered role state predicates O_0, O_1, O_2, O_3 denotes O's state during the exchange subprotocol. O_0 is the state of O at the start of the protocol. O_1 corresponds to the state in O has sent, me_1, its PCS intended for R. In transition from O_0 to O_1, O sets its abort timer, Z_{ab}. In state O_1, now waits for me_2, R's PCS and uses Z_{ab}, to decide how long to wait. If Z_{ab} times out, O_0 may request T to abort the exchange by sending ma_1 and transit to $O_{ab?}$. If however, the O sees R's PCS on the network and Z_{ab} has not timed out, O may read the PCS and transit to O_2. After reading the commitment, O now sends its signature, me_3 to R and transits to state O_3. When O does this, it sets its resolve timer Z_{res} to decide how long to wait for R's response. If Z_{res} times out, O may request T for a resolution by sending

mr_1, and transit to state $O_{res?}$. If however, O sees R's signature on the network and Z_{res} has not timed out, O may read it and go to state O_{com}. $O_{ab?}$ corresponds to the state in which O has issued an abort request to T, and O_{ab1} corresponds to the state in which O has an abort_token for this request and O_{res1} corresponds to the state in which it receives a resolution for this request. $O_{res?}$ corresponds to the state in which O has issued a resolve request to T and O_{ab2} and O_{res2} correspond to the states in which O has received an abort_token or a resolution, respectively for this request.

The theory $\mathbf{O_{timeouts}}$ is given in table 4.2. There are two timeout rules, one for Z_{ab} and one for Z_{res}.

O, role theory for O:

O_{1} : $O_0(pd), Z_{ab}(k_o, unset) \rightarrow O_1(pd, me_1), N(me_1), Z_{ab}(k_o, set)$

$O_{ab?}$: $O_1(pd, me_1), Z_{ab}(k_o, timed\_out) \rightarrow$

$O_{ab?}(pd, me_1, ma_1), P_n(k_o, k_t, ma_1), Z_{ab}(k_o, timed\_out)$

O_{2} : $O_1(pd, me_1), N(me_2), Z_{ab}(k_o, set) \rightarrow O_2(pd, me_1, me_2), Z_{ab}(k_o, set)$

O_{3} : $O_2(pd, me_1, me_2), Z_{res}(k_o, unset) \rightarrow$

$O_3(pd, me_1, me_2, me_3), N(me_3), Z_{res}(k_o, set)$

$O_{res?}$: $O_3(pd, me_1, me_2, me_3), Z_{res}(k_o, timed\_out) \rightarrow Z_{res}(k_o, timed\_out),$

$O_{res?}(pd, me_1, me_2, me_3, mr_1), P_n(k_o, k_t, res\_req)$

O_{com} : $O_3(pd, me_1, me_2, me_3), N(me_4), Z_{res}(k_o, set) \rightarrow$

$O_{com}(pd, me_1, me_2, me_3, me_4), Z_{res}(k_o, set)$

O_{ab1} : $O_{ab?}(pd, me_1, ma_1), P_n(k_o, k_t, ab\_tok) \rightarrow O_{ab1}(pd, me_1, ma_1, ab\_tok)$

O_{res1} : $O_{ab?}(pd, me_1, ma_1), P_n(k_o, k_t, res\_cn) \rightarrow O_{res1}(pd, me_1, ma_1, res\_cn)$

O_{ab2} : $O_{res?}(pd, me_1, me_2, me_3, mr_1), P_n(k_o, k_t, ab\_tok) \rightarrow$

$O_{ab2}(pd, me_1, me_2, me_3, mr_1, ab\_tok)$

O_{res2} : $O_{res?}(pd, me_1, me_2, me_3, mr_1), P_n(k_o, k_t, res\_cn) \rightarrow$

$O_{res2}(pd, me_1, me_2, me_3, mr_1, res\_cn)$

Table 4.1: Role theory for O with timers

$\mathbf{O_{timeouts}}$, timeout rules for O:

$$Z_{ab}: \quad Z_{ab}(k_o, set) \rightarrow Z_{ab}(k_o, timed\_out)$$

$$Z_{res}: \quad Z_{res}(k_o, set) \rightarrow Z_{res}(k_o, timed\_out)$$

Table 4.2: The timeout rules for O

Chapter 5

Protocol properties

In this chapter, we formalize properties that are desired of signature-exchange protocols using the game-theoretic framework of chapter 4.

Recall that a protocol definition consists of a set of possession rules, a protocol theory, say $\mathbf{P}$, two theories that model dishonest behavior of signers, and an initial set of facts. For the rest of this chapter, we pick a protocol definition, and fix it. Let $\mathbf{P} = \mathbf{O} \uplus \mathbf{R} \uplus \mathbf{T} \uplus \mathbf{O_{timeouts}} \uplus \mathbf{R_{timeouts}}$ be the protocol theory for signers O and R with public keys k_o and k_r and trusted third party T with public key k_t. $\mathbf{O}$ and $\mathbf{R}$ are role theories for O and R with public keys k_o and k_r respectively. $\mathbf{T}$ contains the role theory for T along with the timeout rules of the timers of T. Let S_0 be the initial set of facts, $\mathbf{O_{threat}}$ and $\mathbf{R_{threat}}$ be the set of dishonest rules for O and R respectively.

Let A be one of the signers, O or R, and B be the other signer. If A is O then let $\mathbf{A}$ be $\mathbf{O}$, $\mathbf{A_{timeouts}}$ be $\mathbf{O_{timeouts}}$, $\mathbf{A_{threat}}$ be $\mathbf{O_{threat}}$, and $TTPchannel_A$ be P_n, and k_a be k_o. If A is R then let $\mathbf{A}$ be $\mathbf{R}$, $\mathbf{A_{timeouts}}$ be $\mathbf{R_{timeouts}}$, $\mathbf{A_{threat}}$ be $\mathbf{R_{threat}}$, $TTPchannel_A$ be S_n, and k_a be k_r. $\mathbf{B}$, $\mathbf{B_{timeouts}}$, $\mathbf{B_{threat}}$, k_b and $TTPchannel_B$ are similarly defined. We start by defining fairness.

5.1 Fairness

Fairness is the basic symmetry property desired of an exchange protocol: either both signers get each other's signature or none does. The famous impossibility result [24, 42] demonstrates 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. To avoid T becoming the communication bottleneck, modern fair exchange protocols [3, 28, 38] use T optimistically (see section 5.3). We define fairness using the notion of strategy (see definition 4.5.3). We will show that this definition is equivalent to a common definition of fairness in terms of state reachability (chapter 3, [28, 15]).

Definition 5.1.1. A protocol is *fair* for honest A if, for every state S reachable by honest A such that (dishonest) B possesses A's signature on the pre-agreed text in S, and for each bound on the number of moves that a dishonest B makes, the coalition of A and T has an honest strategy from S to give A B's signature.

Fairness requires that once B has A's signature, then an honest A should *always* be able to obtain B's signature on the pre-agreed text regardless of B's moves. We say that a state is *potentially successful* for A if A *may* possibly get B's signature in S with the help of T:

Definition 5.1.2. A state S reachable for honest A is *potentially successful* for an honest A if there is a finite trace tr from S such that:

1. tr terminates in a state successful for A, and
2. each transition rule in tr is labeled by a rule in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}}$.

Please note that this should not be confused with fairness in which we require that A *always* be able to obtain B's signature on the pre-agreed text regardless of B's moves.

Timers of B do not affect actions of A and T. We can use this fact to show (proposition 5.1.4) a state S is potentially successful for honest A, even if there is a trace from S that involves the transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T} \cup \mathbf{B_{timeouts}}$ and ends in a state successful for B. In other words, a timeout of B does not enable A's ability to contact T and get B's signature.

The proof of proposition 5.1.4 uses proposition 5.1.3. Proposition 5.1.3 follows from the fact that in an interleaving semantics of concurrency we can commute application of transition rules that involve independent parts of the system (proposition 4.0.39).

Proposition 5.1.3. *Let S, S' be reachable states such that S' be obtained from S by an application of $t_1 \in \mathbf{B_{timeouts}}$ followed by an application of $t_2 \in \mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$. We can commute the order of application of t_1 and t_2,* i.e., *S' can also be obtained from S by the application of t_2, followed by the application of t_1.*

Proof. The timer predicates of B do not occur in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$. Therefore, t_1 and t_2 affect independent parts of S and by proposition 4.0.39, we can commute the application of t_1 and t_2. □

Now, we show that a timeout of B does not enable the capability of A to contact T and get B's signature.

Proposition 5.1.4. *Suppose there is a trace tr from S that uses only transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T} \cup \mathbf{B_{timeouts}}$ and ends in a state successful for A. Then S is potentially successful for A,* i.e., *there is a trace that uses only rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$ and ends in a state successful for A.*

Proof. Intuitively, timeouts of B does not affect the ability of A to contact T.

Let tr be a trace from S that uses only transition rule in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T} \cup \mathbf{B_{timeouts}}$ and ends in a state successful for A. By repeated application of proposition 5.1.3, we can push the timeouts of B towards the end and get a trace tr' from S such that

i) tr' ends in a state successful for A, and

ii) tr' consists of a trace that uses only transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$ followed by the timeout rules of timers of B.

By proposition 4.4.1, timeouts of B does not affect A's possession. Hence, there is a trace from S that uses only transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$ and ends in a state successful for A. $\square$

Now we state fairness in terms of reachability:

Proposition 5.1.5. *A protocol is fair for honest A if and only if, $\forall$ states S reachable for honest A such that B has A's signature on the pre-agreed text in S, S is potentially successful for A.*

Proof. ($\Rightarrow$) Intuitively, if after obtaining A's signature B takes no further action, then a fair protocol must provide some means for A to obtain B's signature.

Suppose that the protocol is fair for honest A, and S is a state reachable for honest A, such that B has A's signature in S. Assuming that dishonest B makes no moves, let *ctr* be the tree of all possible traces at S. Now, since B makes no moves, each edge in *ctr* must be labeled by a rule in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}} \cup \mathbf{B_{timeouts}}$. The coalition of A and T controls all the edges labeled by rules in $\mathbf{A} \cup \mathbf{T}$. If E is selection of edges under the control of A and T, the tree $ctr \backslash E$ is obtained from *ctr* by removing all the edges in E along with its descendents. $ctr \backslash E$ 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. That is there is a selection of edges E such that each of the leaf node of the tree $ctr \backslash E$ is labeled by a node successful for A. Now pick one such selection, say E and one of leaf node in $ctr \backslash E$. Consider the trace from the root to leaf node. This trace ends in a state successful for A and each transition is labeled by a rule in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}} \cup \mathbf{B_{timeouts}}$. By 5.1.4, S is potentially successful for A.

($\Leftarrow$) Intuitively, if B has A's signature then B continues to hold the signature. Since every state S' in which B has A's signature is potentially successful for A, the coalition of A and T may (safely) choose to do any of the actions available to it in such a state. If in a state A and T run out of possible actions, then A must have B's signature (this state is also potentially successful for A).

Suppose S is a state reachable for honest A such that B has A's signature on the pre-agreed text in S. Let l be the bound on the number of moves made by dishonest B. The coalition of A and T controls all the edges labeled by rules in $\mathbf{A} \cup \mathbf{T}$. If E is selection of edges under the control of A and T, the tree $ctr \backslash E$ is obtained from ctr by removing all the edges in E along with its descendents. $ctr \backslash E$ is a strategy for the coalition of A and T.

In order to show that the protocol is fair, we need to show that there is a section of edges, E under A's control such that in $ctr \backslash E$:

i) each leaf node is successful for A, and

ii) in each leaf node all timers are timed out.

Let E be the empty set and consider the strategy $ctr \backslash E = ctr$. Consider a leaf node, N, of this strategy and let N be labeled by S'. Since B has A's signature in S, B has A's signature in S'(proposition 4.4.1). Therefore S' is potentially successful for A.

Therefore there is a trace, tr from S' with the edges labeled by transition rules in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}}$ leading to a state successful for A. Since ctr is full continuation tree, tr must be empty. Therefore S' is successful for A. Since N was an arbitrary node in ctr, we get ctr is a strategy of the coalition of A and T to give A B's signature. $\square$

For the rest of the thesis, we shall limit our attention to protocol theories that are fair for honest signers.

5.2 Completeness and effectiveness

If both the signers are honest, then they should be able to exchange signatures by using the protocol, otherwise the protocol would not be very useful. We formulate this property of the protocol as *completeness*.

Definition 5.2.1. A protocol is **complete** if, assuming both A *and* its counterparty B are honest, there is a trace from S_0 to a state, successful for both A and B.

An important design goal is to ensure that an honest A should not be left hanging indefinitely, and A should be able to force a timely termination. We formulate this property as *effectiveness*. In order to formulate this property we need the following definition:

Definition 5.2.2. Let $\mathbf{C}$ be a role theory for participant C with public key k_c.

1. A role state predicate of C is said to be *final for C* if in $\mathbf{C}$, this role state occurs only on the right of the state transition rules.

2. A state S is *final for an honest C* if S contains an occurrence of a role state predicate that is final for C.

A protocol is said to be effective for an honest A, if an honest A can with the help of T always reach a state final for honest A. The following definition is motivated by [3].

Definition 5.2.3. A protocol is **effective for honest** A if, for any state S reachable for honest A, there is a tr from S such that

1. each edge in tr is labeled by a transition in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}}$, and

2. tr ends in a state final for A.

5.3 Optimistic protocols

In literature [28, 1, 3], a protocol is said to be optimistic, if there is a trace such that the signers can exchange signatures without contacting the trusted third party. However, this definition does not capture an inherent assumption: the signers should be patient for the other signer for this optimistic exchange to take place. We will say that a protocol is optimistic for a signer, Alice, if when Alice is willing to wait "long enough", then the other signer, Bob, has a strategy to exchange signatures without any of them contacting the third party. Optimistic protocols have been a subject of extensive research, since they potentially provide a practical means of fair exchange between mistrusting agents without relying on the trusted third party in most instances.

We say that A does not send a message to T in the transition from S to S' if

1. The transition is an application of a rule in $\mathbf{A} \cup \mathbf{A_{threat}}$, and
2. No fact created by the transition matches a term in the left hand side of a rule in $\mathbf{T}$.

We may now formulate optimistic protocols.

Definition 5.3.1. A fair protocol is *optimistic for A* if, assuming B is honest and A controls the timeouts of both A and B, A has an honest strategy from the initial state, S_0, such that

1. no messages are sent by any signer to T in this strategy, and
2. A exchanges signatures with B, *i.e.*, each leaf node is labeled by a state successful for both A and B.

Any trace from the root to the leaf node in this strategy is an *optimistic trace for A*. A protocol is *optimistic* if it is optimistic for both signers.

We shall assume that if a protocol is optimistic for both A and B, then in the definition above the trees obtained as a result of strategies are the same for both A and B. Therefore an optimistic trace for A is also an optimistic trace for B, and vice-versa. We shall henceforth refer to these as *optimistic traces* and not distinguish between them.

Please note that our definition implies that both A and B uses timers to contact T, if they are following the protocol. In the definition above, B's willingness to wait "long enough" is modeled by giving A the ability to signal B when B has the option of contacting T.

Hence a protocol is optimistic for A if, whenever B is willing to wait "long enough" , the signature exchange may take place without any signer contacting T if A so desires. More precisely, there is a schedule of timeouts such that A may choose to exchange signatures with B without either of them needing to go to T.

Now if A has the controls the time out of A and B, then A has a strategy from S_0 to exchange signatures with honest B. If a state S occurs as a result of this strategy, then A also has a strategy from S to exchange signatures with B:

Proposition 5.3.2. *In an optimistic protocol, if A controls the timeouts of both A and B, then an honest A has a strategy from each point in an optimistic trace to exchange signatures with honest B.*

Proof. Let tr be an optimistic trace and states in tr be $S_0, S_1, \ldots, S_n$ in that order.

Intuitively, since the protocol is optimistic, if A controls the timeouts of both A and B, then honest A has a strategy from S_0 to exchange signatures with honest B. Each S_i is realized as a result of this strategy. In order to accomplish the exchange from S_i for $i > 0$, A just continues with the strategy.

Assuming A and B are honest, let ctr, ctr_i be the trees of all possible traces from S_0 and S_i respectively. Clearly ctr_i is a subtree of ctr.

In the trees, A controls all edges that are labeled by transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{B_{timeouts}}$. If E is a selection of edges under A's control in ctr, $ctr \backslash E$ is the tree derived from ctr by deleting each edge in E along with its descendants. Since A has a strategy to exchange signatures at S_0, there is a selection of edges, say E, under A's control in ctr such that in the tree $ctr \backslash E$:

i) each leaf node is successful for A and B, and

ii) all timers are timed in the leaf nodes.

Fix such a selection, E_0.

In order to continue with the same strategy at S_i also, A selects all the edges in ctr_i that occur in E_0. Now, let E_i be the subset of edges in ctr_i that occur in E_0 also. Each edge in E_i is under A's control. Now clearly, $ctr_i \backslash E_i$ is the subtree of $ctr \backslash E_0$ that is rooted at the node labeled as S_i. Hence in the tree $ctr \backslash E_i$:

i) each leaf node is successful for A and B, and

ii) all timers are timed in the leaf nodes.

Therefore A has a strategy from S_i to exchange signatures with B. $\square$

5.4 Signer bias and advantage

Intuitively, a protocol is fair (see section 5.1) if either both parties exchange signatures, or none do. Basic fairness is not always sufficient, however. In scenarios such as online trading and auction bidding where resource commitment is important, a signer's ability to *choose* the outcome can be almost as important as the outcome itself as explained in chapter 1. Even if the final transaction is binding on both parties (and thus fair), the ability to unilaterally decide *whether* the transaction happens or not can be of great value. We formalize this ability using the concept of *advantage* (see definition 5.4.1).

5.4.1 Signer bias

Our definition of optimistic protocols (see definition 5.3.1) implies that honest signers use timers to contact T, and if a signer A is willing to wait "long enough", then signature exchange may take take place without any signer contacting T if B so desires.

Honest signers are simply guaranteed to follow the protocol. Mere honesty, however, does not capture all the nuisances of the natural behavior of signers in most commercial transactions. Honest signers typically have an inherent bias towards the positive outcome of the protocol (otherwise, they would not engage in it). We will say that an honest signer A is *interested* if, whenever it is permitted by the protocol specification to contact T immediately with an abort request, it waits for a "long"

Signer bias	Behavior at decision points
Honest	Waits or contacts T immediately
Interested	Waits before contacting T to abort
Optimistic	Waits before contacting T

Table 5.1: Signer Bias

time before asking T to abort the exchange.

Moreover, they may be optimistic. They prefer to attempt to resolve the transaction amicably, without involving T, and only use T as the last resort. We will say that an honest signer A is *optimistic* if, whenever it is permitted by the protocol specification to contact T immediately, it waits for a "long" time before contacting T the exchange.

The precise meaning of "long," as will be explained in section 5.4.2, is "long enough to ensure that B's messages to T, if any, arrive before A's message." As discussed in section 1, we *must* consider biased signers when analyzing properties of signature exchange protocols. The difference between signer biases is summarized in table 5.1.

For example, consider the behavior of responder R in the GJM protocol with timers (see section 4.6) immediately after it responds to O's first message with its own PCS. While the protocol specification permits R to contact the T immediately with a resolve request, in reality R may be optimistic, *i.e.*, it may prefer to resolve the protocol amicably by normal exchange with O instead of resorting to T as soon as it has the opportunity to do so. Therefore, after sending its PCS to O in the second message of the protocol, R will wait for O's response for a long time before resorting to T with a resolve request. If however, R is interested, it will proceed to contact T with a resolve request.

5.4.2 Advantage

We now formalize a signer's ability to control the outcome of a protocol. For this section assume that A is honest and B dishonest. This definition is the generalization of definition 3.3.2 in chapter 3.

Definition 5.4.1. Let S be a reachable state.

1. B has the *power to abort at* S if B has a strategy from S to prevent A from obtaining B's signature with some finite number of (possibly dishonest) moves.

2. B has the *power to obtain* A*'s signature at* S if B has a strategy to obtain A's signature, with some finite number of (possibly dishonest) moves.

3. B has the *advantage* over A at S if B has both the power to abort at S and the power to obtain A's signature at S.

Definition 5.4.2. A protocol is *balanced for* A if for all reachable states S, B does not have an advantage over A at S.

In optimistic protocols, the propensity of a biased signer to wait for a long time before contacting T can be exploited by his opponent. If A is optimistic, B will enjoy the ability to contact T before A times out and contacts T. We will model optimistic bias of A by giving B the ability to schedule timeout rules of A. In reality, of course, the misbehaving signer does not actually set other parties' timeouts. This is simply an abstract representation of the fact that A's timeouts are optimistic, *i.e.*, "long enough" to effectively relinquish A's control over message scheduling and give a dishonest B the opportunity to contact T before A does. If B wants to schedule its messages to T ahead of A, it will delay the timeout rules of A's (again, this is just an abstraction of the fact that the timeouts of an optimistic A are long). If B wants A to contact T first, it will set off the timeout rule of A's timer before sending anything to T.

If A is interested, B will enjoy the ability to contact T before A times out and quits the exchange or contacts T with a request to abort the protocol. We will model interested bias of A by giving B the ability to schedule timeout rules of abort timers of A.

Definitions 5.4.1 and 5.4.2 can thus be extended to cases where A is interested or optimistic by permitting B to have control over some of timeouts. The table 5.2 extends definition 5.4.1 to biased signers. From the above table, it is clear that if the protocol provides a signer with the advantage against honest opponents, it also has the advantage against optimistic and interested opponents. Advantage against interested opponents implies advantage against optimistic opponents.

A's bias	B controls…
Honest	$\mathbf{B}$, $\mathbf{B_{threat}}$, $\mathbf{B_{timeouts}}$
Interested	$\mathbf{B}$, $\mathbf{B_{threat}}$, $\mathbf{B_{timeouts}}$, $\mathbf{A_{atimeouts}}$
Optimistic	$\mathbf{B}$, $\mathbf{B_{threat}}$, $\mathbf{B_{timeouts}}$, $\mathbf{A_{timeouts}}$

Table 5.2: Signer bias and control of timeouts

Chapter 6

Impossibility of balance

In this chapter we shall study relationships between fairness, effectiveness, optimism and advantage. We shall show that if a protocol is fair and effective for honest signers, then it must be balanced for honest signers. However, we shall show that completely eliminating advantage is *impossible* in an optimistic two-party signature exchange protocol. Any fair and optimistic protocol must necessarily have a reachable state in which one of the signers enjoys the advantage against an optimistic counterparty.

Recall that a protocol definition consists of a set of possession rules, a protocol theory, say **P**, two theories that model dishonest behavior of signers, and an initial set of facts. For the rest of this chapter, we pick a protocol definition, and fix it. Let $\mathbf{P} = \mathbf{O} \uplus \mathbf{R} \uplus \mathbf{T} \uplus \mathbf{O_{timeouts}} \uplus \mathbf{R_{timeouts}}$ be the protocol theory for signers O and R with public keys k_o and k_r and trusted third party T with public key k_t. **O** and $\mathbf{R}$ are role theories for O and R with public keys k_o and k_r respectively. $\mathbf{T}$ contains the role theory for T along with the timeout rules of its timers. Let S_0 be the initial set of facts, $\mathbf{O_{threat}}$ and $\mathbf{R_{threat}}$ be the set of dishonest rules for O and R respectively. For the rest of this chapter, we shall assume that the protocol is fair.

Let A be one of the signers, O or R, and B be the other signer. If A is O let $\mathbf{A}$ be $\mathbf{O}$, $\mathbf{A_{timeouts}}$ be $\mathbf{O_{timeouts}}$, $\mathbf{A_{threat}}$ be $\mathbf{O_{threat}}$, and $TTPchannel_A$ be P_n, and k_a be k_o. If A is R let $\mathbf{A}$ be $\mathbf{R}$, $\mathbf{A_{timeouts}}$ be $\mathbf{R_{timeouts}}$, $\mathbf{A_{threat}}$ be $\mathbf{R_{threat}}$, $TTPchannel_A$ be S_n, and k_a be k_r. Similarly, we can define $\mathbf{B}, \mathbf{B_{timeouts}}, \mathbf{B_{threat}}$, k_b and $TTPchannel_B$.

6.1 Balance and effectiveness

We establish that for single runs, if a protocol is fair and effective for an honest signer, then it is balanced for the signer.

Proposition 6.1.1. *If a protocol is fair and effective for honest A, then it is balanced for honest A.*

Proof. Intuitively, a dishonest B can only prevent scenarios that are a result of the actions that B can control. If the protocol is effective for honest A, it is possible for an honest A to contact T and get a fair completion. Since A is honest and not biased, the timeout of A is not "long" enough for B to schedule its own messages ahead of A. Hence, if A non-deterministically chooses to contact T and gets a particular outcome (*i.e.*, abort or successful exchange), B will not be able to force the other outcome. For a precise proof, we recall the definition of advantage.

Let S be a state reachable by honest A. Assuming that A is honest, let ctr be the tree of all possible traces from S_j. An edge in ctr is under B's control only if it is labeled by a transition in $\mathbf{B} \cup \mathbf{B_{threat}} \cup \mathbf{B_{timeouts}}$. Given a selection E of edges under B's control, $ctr\backslash E$ is the tree derived from ctr by deleting each edge in E along with its descendants. The selection E represents actions that A considers undesirable, and $ctr\backslash E$ represents all the possible traces if transitions in E do not take place. B has the power to obtain A's signature if there is a selection, E, such that each leaf node in $ctr\backslash E$ is labeled by a state successful for A. B has the power to abort if there is a selection of edges E under B's control such that all the nodes in the truncated tree $ctr\backslash E$ are unsuccessful for A. B has the advantage if it has both the power to abort and power to obtain A's signature.

If the protocol is effective for honest A, then there is a trace, tr, from S such that

1. each edge in the trace is labeled by a transition in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$, and

2. tr ends in a node labeled by a state, S', final for A.

The theories $\mathbf{B}, \mathbf{B_{threat}}$ and $\mathbf{B_{timeouts}}$ are disjoint from $\mathbf{A}, \mathbf{A_{timeouts}}$ and $\mathbf{T}$. Therefore no edge in tr is under B's control. Therefore, for every possible selection E of edges under A's control, $ctr\backslash E$ contains tr and in particular, a node labeled by S'.

Now if S' is successful for A, then B does not have power to abort. If S'

is unsuccessful and final for A, then A does not have B's signature in any state reachable from S'. Therefore, in each leaf node of the subtree of a strategy $ctr\backslash E$ rooted at S', A does not have B's signature. By fairness, B also does not have A's signature in these nodes. Hence, if S' is unsuccessful for A, B does have power to obtain A's signature.

Hence if a protocol is fair and effective for A, then it is balanced for A. $\square$

We see that if a protocol is fair and effective for signers, then it is balanced for them. The reason for this is that honest signers may choose to exercise any option available to it, including contacting T. We shall see, however, that if the signers have a natural bias, then the protocol must give an advantage to some signer.

6.2 A three-valued function

In order to establish the impossibility result, we define a three-valued function on states reachable for honest signers.

For the rest of this section, we shall assume that the protocol is fair.

Recall that a state S is potentially successful for A if A *may* obtain B's signature on the pre-agreed text with the help of T (see definition 5.1.2), *i.e.*, there is a trace from S involving only transition rules in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}}$ leading to a state successful for A.

Now we define a three-valued function win_A for all states S reachable for honest A as follows:

$$\begin{aligned} win_A(S) \quad &= 2, \quad \text{if } S \text{ is successful for honest } A \\ &= 1, \quad \text{if } S \text{ is unsuccessful, but potentially successful for } A \\ &= 0, \quad \text{otherwise.} \end{aligned}$$

We can define win_B on states reachable by honest B similarly.

We assume that $win_A(S_0) = win_B(S_0) = 0$, where S_0 is the initial set of facts. This is a reasonable assumption, since neither A nor B have sent any information in S_0. Now, we state and prove some properties of win_A and win_B.

Proposition 6.2.1. *For all states S reachable for honest A and honest B, $(win_A(S), win_B(S))$ never takes the value $(0, 2)$ or $(2, 0)$.*

Proof. If A possesses B's signature at S and B is honest, then by fairness B may get A's signature with the help of T, *i.e.*, S must be potentially successful for B (proposition 5.1.5). Therefore, if $win_A(S) = 2$, then $win_B(S) \neq 0$. Similarly, if $win_B(S) = 2$, then $win_A(S) \neq 0$. □

The following proposition says that says that the value of win_A cannot change from 0 to 2 without going through 1:

Proposition 6.2.2. *Let S, S' be states reachable for honest A such that S' is obtained from S. If $win_A(S) = 0$, then $win_A(S') \neq 2$.*

Proof. We are given $win_A(S) = 0$. Assume that $win_A(S') = 2$. Hence A does not possess B's signature at S ($win_A(S) = 0$), and A possesses the signature in S' ($win_A(S) = 2$).

Since the possession of a participant changes only by its own actions (proposition 4.4.1), the transition from S to S' must use a rule in $\mathbf{A} \cup \mathbf{A_{threat}}$. Furthermore, since A is honest this rule must be in $\mathbf{A}$.

Now the transition from S to S' is in $\mathbf{A}$ and S' is successful for A. Therefore, by definition, S is potentially successful for A, *i.e.*, $win_A(S) \neq 0$. This contradicts the fact $win_A(S) = 0$. Therefore, $win_A(S') \neq 2$. □

The above proposition implies that the value of win_A must change from 0 to 1 before it changes to 2. We shall now establish that if the value of win_A changes from 0 to 1 then it must be because of a move of B. Recall that a state S is potentially successful for A, if there exists a trace from S that uses only transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$ and ends in a state successful for A. In fact, we showed in proposition 5.1.4 that S is potentially successful for A even if there is a trace from S leading to a state successful for A that uses transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T} \cup \mathbf{B_{timeouts}}$.

Proposition 6.2.3. *Let S, S' be states reachable for honest A, such that S' is obtained from S by using a transition rule $t\ \sigma$. If $win_A(S) = 0$ and $win_A(S') = 1$, then the following hold:*

1. *$t \in \mathbf{B} \cup \mathbf{B_{threat}}$.*
2. *A new message is deposited on the network or on one of the channels to T,* i.e.*, there is a fact $N(m)$ or $TTPchannel_A(k_b, k_t, m)$ or $TTPchannel_B(k_b, k_t, m)$*

that occurs in S' and not in S, where m is a ground term of the sort mssg.

Proof. Intuitively, it is possible at S' but not at S for A to contact T and get B's signature at S. Hence, the transition from S to S' cannot be a action of A, T or a timeout of A. Moreover, we see from proposition 5.1.4 that timeouts of B also does not affect the ability of A to contact T and get B's signature. Hence, the transition must be a action of B. Also, this move should have enabled some rule of A or T that was not enabled before. The only way a B action can enables a rule of A or T is if some message is sent to them.

More precisely since A is honest, t must be in one of the following: $\mathbf{A}$, $\mathbf{B}$, $\mathbf{T}$, $\mathbf{B_{threat}}$, $\mathbf{A_{timeouts}}$, or $\mathbf{B_{timeouts}}$.

Since $win_A(S') = 1$, there is a trace, tr, from S' that involves the transition rules in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$ leading to state successful for A. If $t \in \mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}} \cup \mathbf{B_{timeouts}}$, then we can extend tr to get a trace using transition in $\mathbf{A} \cup \mathbf{T} \cup \mathbf{A_{timeouts}} \cup \mathbf{B_{timeouts}}$ leading to a state successful for A. By proposition 5.1.4, S is also potentially successful for A. That contradicts the fact that $win_A(S) = 0$, and hence $t \in \mathbf{B} \cup \mathbf{B_{threat}}$.

Since S is not potentially successful and S' is potentially successful, there must be a fact in S' that is not present in S. Furthermore, this fact must enable some rule in $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$ in the trace tr that was not enabled before. The only common predicates in $\mathbf{B} \cup \mathbf{B_{threat}}$ and $\mathbf{A} \cup \mathbf{A_{timeouts}} \cup \mathbf{T}$ are $N, TTPchannel_A$, and $TTPchannel_B$. Hence, we obtain that there is some ground term m of the sort

$mssg$ such that a fact $N(m)$ or $TTPchannel_A(k_b, k_t, m)$ or $TTPchannel_B(k_b, k_t, m)$ occurs in S' and not in S. $\square$

Now, if win_A changes from 1 to 2, it must be a result of an action of A reading something from the network or the TTP channel (since A's possession increased). Furthermore, we show that as a consequence of fairness, win_B must be non-zero to begin with in that case:

Proposition 6.2.4. *Let S, S' be states reachable for honest A and B such that S' is obtained from S by using a transition rule t. If $win_A(S) = 1$ and $win_A(S') = 2$, then $t \in \mathbf{A}$. Furthermore, $win_B(S) \neq 0$.*

Proof. A does not possess B's signature in S, but possesses it in S'. Therefore the transition must be A receiving some message (proposition 4.4.1). Therefore, $t \in \mathbf{A} \cup \mathbf{A_{threat}}$. A is honest, therefore $t \in \mathbf{A}$.

By proposition 4.4.1, in the transition from S to S', A reads some message. Now, since we have separated the receive transitions from send transitions in the role theory $\mathbf{A}$, no new messages are deposited on the network or TTP channels in the transition from S to S'.

It is an immediate consequence of fairness that since $win_A(S') = 2$, $win_B(S') \neq 0$ (proposition 6.2.1). Therefore, if $win_B(S) = 0$, then by proposition 6.2.2, $win_B(S')$ $= 1$. By proposition 6.2.3, some new message must been deposited on the network or a TTP channel in the transition from S to S' by A. This contradicts the fact that no new messages are deposited on the network or TTP channels in the transition from S to S'. Hence $win_B(S) \neq 0$. $\square$

6.3 Balance in optimistic protocols

In this section, we show that completely eliminating advantage is *impossible* in an optimistic two-party signature exchange protocol. Any fair, optimistic protocol must necessarily have a reachable state S in which one of the signers enjoys the advantage against an optimistic counterparty The proof is a three-valued version of Even's impossibility argument [24] for the two-party case.

Recall that a state S is potentially successful for B, if there exists a trace from S leading to a state successful for B such that every edge in the trace is labeled by a transition rule in $\mathbf{B} \cup \mathbf{B_{timeouts}} \cup \mathbf{T}$. Recall the definition of win_A.

$$\begin{aligned} win_A(S) \quad &= 2, \quad \text{if } S \text{ is successful for } A \\ &= 1, \quad \text{if } S \text{ is unsuccessful, but potentially successful for } A \\ &= 0, \quad \text{otherwise.} \end{aligned}$$

For the rest of this section we shall assume that our signature exchange protocol is fair and optimistic. Recall that S_0 is the initial set of facts. Since the protocol is optimistic, there exists an optimistic trace tr from S_0 to a final state which is successful for both A and B (see definition 5.3.1). Let the nodes in tr be labeled as $S_0, S_1, \ldots, S_n$. We have $win_A(S_0) = win_A(S_0) = 0$, and $win_A(S_n) = win_A(S_n) = 2$

(since S_n is successful for A and B).

Proposition 6.3.1. *If* $(win_A(S_k), win_B(S_k)) = (0,0)$*, then* $(win_A(S_{k+1}), win_B(S_{k+1})) \neq (1,1)$*.*

Proof. By proposition 6.2.3, if $win_A(S_{k+1}) = 1$, then the transition from S_k to S_{k+1} is labeled by a rule in $\mathbf{B} \cup \mathbf{B_{threat}}$. Since tr is an optimistic trace, the edge is labeled by a rule in $\mathbf{B}$. Similarly if $win_B(S_{k+1}) = 1$, the edge is labeled by a rule in $\mathbf{A}$. Since $\mathbf{A}$ and $\mathbf{B}$ are disjoint, we get $(win_A(S_{k+1}), win_B(S_{k+1})) \neq (1,1)$. □

The proof of the above proposition might suggest that the proposition is an artifact of our model and may not be true in a model of concurrency where a single step may involve simultaneous actions by independent agents. Nevertheless, a form of the proposition and the following lemma will still be true. This is because the value of *win* of a signer changes from 0 to 1 only if the counterparty acts. Hence a dishonest A can prevent change in the value of win_B, and vice-versa. Hence, we can always get a state (not necessarily in the optimistic trace) where a form of the following lemma holds:

Lemma 6.3.2 (Existence of advantage point). *If* $(win_A(S_i), win_B(S_i)) = (0,0)$*, then* $\exists\ j > i$ *such that* $(win_A(S_j), win_B(S_j))$ *is either* $(0,1)$ *or* $(1,0)$*.*

Proof. Since tr is optimistic, $(win_A(S_n), win_B(S_n)) = (2,2)$. Let j be the smallest $l > i$ such that $(win_A(S_l), win_B(S_l)) \neq (0,0)$. Then $(win_A(S_{j-1}), win_B(S_{j-1})) = (0,0)$. By proposition 6.2.1, $win_A(S_j) \neq 2$ and $win_B(S_j) \neq 2$. By proposition 6.3.1,

$(win_A(S_j), win_B(S_j)) \neq (1,1)$. Hence $(win_A(S_j), win_B(S_j))$ is either $(0,1)$, or $(1,0)$.

$\square$

Recall that a dishonest A has the power to abort if A can prevent B from getting A's signature regardless of the actions of B, T and timeouts of B. We define $PowerAbort_A$ for all states S_i in the optimistic trace, tr, as follows:

$$\begin{aligned} PowerAbort_A(S_i) \quad &= 1, \quad \text{if dishonest } A \text{ has the power to abort against} \\ &\qquad\quad \text{an honest } B \text{ at } S_i, \\ &= 0, \quad \text{otherwise.} \end{aligned}$$

Similarly we can define $PowerAbort_B$. An easy consequence of the definition of power to abort is that if $win_B(S_i)$ takes the value 1, then A does not have power to abort against an honest B.

Lemma 6.3.3. *If $win_B(S_j) = 1$, then $PowerAbort_A(S_j) = 0$.*

Proof. Intuitively, a dishonest A can only prevent scenarios that are a result of the actions that A can control. If $win_B(S_j) = 1$, it is possible for an honest B to contact T and obtain A's signature. Therefore A cannot prevent an honest B from obtaining A's signature. For a precise proof, we recall the definition of A having the power to abort against an honest B.

Assuming that B is honest, let *ctr* be the tree of all possible traces from S_j. An edge in *ctr* is under A's control only if it is labeled by a transition in $\mathbf{A} \cup \mathbf{A_{threat}} \cup \mathbf{A_{timeouts}}$. Given a selection E of edges under A's control, $ctr \backslash E$ is the tree derived from *ctr* by deleting each edge in E along with its descendants. The selection E represents actions that A considers undesirable, and $ctr\backslash E$ represents all the possible traces if transitions in E did not take place. A has the power to abort if there is a selection of edges E under A's control such that all the nodes in the truncated tree $ctr\backslash E$ are unsuccessful for B.

If $win_B(S_j) = 1$, then there is a trace, tr, from S_j such that

1. each edge in the trace is labeled by a transition in $\mathbf{B} \cup \mathbf{B_{timeouts}} \cup \mathbf{T}$, and

2. the trace ends in a node labeled by a state successful for B.

The theories $\mathbf{A}, \mathbf{A_{threat}}$ and $\mathbf{A_{timeouts}}$ are disjoint from $\mathbf{B}, \mathbf{B_{timeouts}}$ and $\mathbf{T}$. Therefore no edge in tr is under A's control. Therefore, for every possible selection E of edges under A's control, $ctr\backslash E$ contains tr. Since tr ends in a node successful for B, A does not have power to abort against an honest B. $\square$

Now, we shall establish that if $win_B(S_j) = 0$, then A does have the power to abort against an honest B. In order to show this recall that a state S is potentially successful for B, if there exists a trace from S that uses only transition rules in $\mathbf{B} \cup \mathbf{B_{timeouts}} \cup \mathbf{T}$ and ends in a state successful for B. In fact, we showed in proposition 5.1.4 that S is potentially successful for B if and only if there is a trace from S leading to a state successful for B that uses transition rules in $\mathbf{B} \cup \mathbf{B_{timeouts}} \cup \mathbf{T} \cup \mathbf{A_{timeouts}}$. Now, we are ready to show that if $win_B = 0$, then A has the power to abort.

Lemma 6.3.4. *If $win_B(S_j) = 0$, then $PowerAbort_A(S_j) = 1$.*

Proof. Intuitively, a dishonest A can prevent any scenario that is a result of its own action. If $win_B(S_j) = 0$ then B does not have A's signature and cannot obtain it even with the help of T. Therefore, in order for B to get A's signature, A must participate. A can prevent B from getting A's signature by refusing to participate in the protocol, *i.e.*, by not taking any action.

Assuming that B is honest, let ctr be the tree of all possible traces from S_j. Given a set of edges E under A's control, let $ctr \backslash E$ be the tree derived from ctr by deleting each edge in E along with its descendants. A has the power to abort only if there is a selection E of edges under A's control such that

1. all the nodes in the truncated tree $ctr \backslash E$ are unsuccessful for B, and

2. all timers are timed out in each of the leaf nodes of $ctr \backslash E$. (A dishonest A can schedule its timeouts, but cannot prevent them from happening eventually).

The selection E is the set of actions that A considers undesirable.

Since B is honest, each edge in ctr must be labeled by one of the six theories: $\mathbf{A}, \mathbf{A_{threat}}, \mathbf{B}, \mathbf{B_{timeouts}}, \mathbf{T}, \mathbf{A_{timeouts}}$. If A does not want to participate in the protocol, A selects all the edges in the tree that are its own actions. Let E_1 be the set of edges in ctr that are labeled by transitions in $\mathbf{A} \cup \mathbf{A_{threat}}$. Each edge in E_1 in under A's control.

Consider $ctr \backslash E_1$. E_1 consists of all the edges in ctr that are labeled by transitions in $\mathbf{A} \cup \mathbf{A_{threat}}$. Therefore, each edge in $ctr \backslash E_1$ is labeled by a transition in $\mathbf{B} \cup \mathbf{B_{timeouts}} \cup \mathbf{T} \cup \mathbf{A_{timeouts}}$.

Hence if there is a node in $ctr \backslash E_1$ that is labeled by a state successful for B, there is a trace from S_j that ends in a state successful for B and uses only the transitions in $\mathbf{B} \cup \mathbf{B_{timeouts}} \cup \mathbf{T} \cup \mathbf{A_{timeouts}}$. By proposition 5.1.4, this means that S_j is potentially successful for B. By definition, $win_B(S_j) \neq 0$. Therefore, if $win_B(S_j) = 0$, each node in $ctr \backslash E_1$ is labeled by states unsuccessful for B.

Moreover since the selection E_1 does not consist of any timeouts, all timers are timed out in each leaf node in $ctr \backslash E_1$. Hence A has the power to abort against an honest B at S_j and $PowerAbort_A(S_j) = 1$. $\square$

Now, we combine lemmas 6.3.2, 6.3.4 and 6.3.3 to show that there is some point in a fair and optimistic protocol such that exactly one of the signers has the power to abort the exchange.

Theorem 6.3.5. *If a protocol is fair and optimistic, then there is a state S such that $(PowerAbort_A(S), PowerAbort_B(S)) = (1, 0)$ or $(0, 1)$.*

Proof. By proposition 6.3.2, there is a state S_j in the optimistic trace such that $(win_A(S_j), win_B(S_j)) = (0, 1)$ or $(1, 0)$. By propositions 6.3.4 and 6.3.3, $(PowerAbort_A(S_j), PowerAbort_B(S_j)) = (1, 0)$ or $(0, 1)$. $\square$

We will now show that balance is impossible to achieve in a fair and optimistic protocol. More precisely, there is a non-initial point in the optimistic flow where one of the signer can control the outcome of the protocol against an optimistic counterparty. We know that in an optimistic trace, if A controls the the timeouts of

A and B, then A has a a strategy to exchange signatures with B (proposition 5.3.2). Hence, combining propositions 5.3.2 and lemmas 6.3.2 and 6.3.4, we get

Theorem 6.3.6 (Impossibility of balance). *Let tr be an optimistic trace of a fair and optimistic protocol. There is a non-initial state S in tr such that one of the signers enjoys the advantage over its optimistic counterparty at S.*

Proof. We use the existence of advantage point (lemma 6.3.2) to get a non-initial state where the value of *win* for some signer is 0. At this point the counterparty can control the outcome of the protocol.

By lemma 6.3.2, there is a state S_j in tr such that $(win_A(S_j), win_B(S_j)) = (1, 0)$ or $(0, 1)$. Clearly $j > 0$ since $(win_A(S_0), win_B(S_0)) = (0, 0)$.

If $win_B(S_j) = 0$, then by lemma 6.3.4 A has the power to abort against an optimistic B at S_j (an optimistic signer is also an honest signer). If B is optimistic, then A controls the timeouts of both A and B. By proposition 5.3.2, since tr is an optimistic trace, A has a strategy from S_j to obtain optimistic B's signature. Hence A enjoys the advantage over an optimistic B at S_j. Similarly if $win_A(S_j) = 0$, B enjoys an advantage over an optimistic A. $\square$

In related work with John Mitchell, Andre Scedrov and Vitaly Shmatikov [16], we have extended the above theorem:

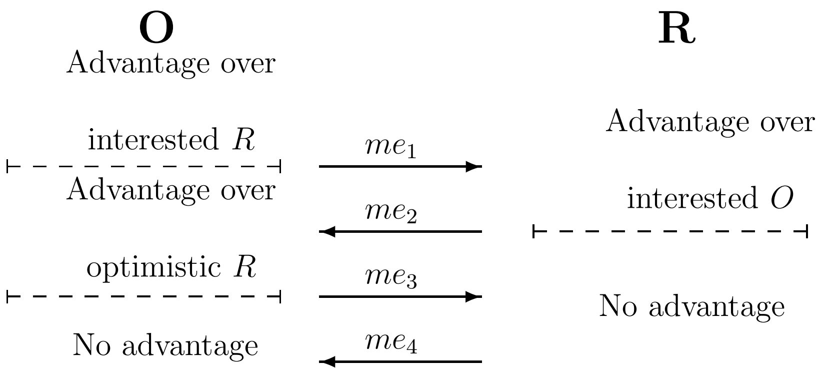
In any fair, timely and optimistic protocol, if a signer A is optimistic then there is a non-initial state at which the counterparty B enjoys an advantage over an optimistic A.

Hence if the protocol is effective in addition to being fair and optimistic *any* optimistic signer must yield an advantage to its counterparty, as opposed to *some* optimistic signer in the theorem 6.3.6 above. Timeliness as defined in [16] is closely related to effectiveness. The proof of this result is outside the scope of this thesis and is not included here.

Since advantage against an optimistic agent cannot be eliminated, the most a protocol can provide an optimistic signer is to prevent the opponent *proving* that it has the advantage. In this sense, the security guarantees provided by abuse-free protocols [3, 28] are the strongest possible. In related work with John Mitchell, Andre Scedrov and Vitaly Shmatikov [16], we use a formal representation of knowledge derived from epistemic logic [33, 25] to formalize the "ability to prove" and analyze abuse-freeness as the lack of *provable advantage*. This work is outside the scope of this thesis and is not included here.

6.4 Advantage flows

We now apply the definitions of advantage defined in section 5.4. We first it on the GJM protocol 4.6 and then apply it on an non-effective optimistic protocol based on the signature exchange protocol given in [9]. In each case, we demonstrate theorem 6.3.6.



O
Advantage over interested R
Advantage over optimistic R
No advantage
me_1
me_2
me_3
me_4
R
Advantage over interested O
No advantage

Figure 6.1: Advantage Flow in GJM protocol

6.4.1 GJM protocol

We illustrate the concepts of advantage developed in section 5.4 by showing how it applies to the Garay-Jakobsson-Mackenzie protocol [28] discussed in chapter 4.6. We can show that the protocol equipped with timers is fair, effective and optimistic for single runs. Figure 6.1 illustrates how advantage of each party decreases as message exchange progresses.

O has the advantage against an interested R until R receives $PCS(k_o, k_r, pd, k_t)$. This is because if O wants to abort, it can simply contact T and obtain an abort token before R has any information that would enable it to request T to resolve. If O wants to get R's signature, O continues talking to R. An interested R would not quit the protocol as long as O continues sending the message. Even after R receives $PCS(k_o, k_r, pd, k_t)$, O still has the advantage against an optimistic R (but not an interested R) until R receives $sig(k_o, pd)$. If R is optimistic, then O has enough time before R times out waiting for $sig(k_o, pd)$ to contact T with an abort request (if O wants to abort) or simply respond to R with $sig_O(m)$ (if O wants to resolve). O does not have the advantage over an interested R, after R receives $PCS(k_o, k_r, pd, k_t)$: O cannot abort since an interested R may contact T and get O's signature.

R has an advantage against an interested O when it receives $PCS(k_o, k_r, pd, k_t)$. If R wants to abort the exchange, it just stops participating in the protocol, and if it wants to obtain an interested O's signature, it continues talking to O using the exchange protocol. In fact, it is this point that is captured by the proof in 6.3.6. Once O receives $PCS(k_r, k_o, pd, k_t)$, R no longer enjoys an advantage, since it cannot force O to abort.

In this protocol both O and R enjoy advantage at some non-initial point in the protocol. We also studied the protocol in [3]. The advantage flow in that protocol and in this protocol is very similar and hence we are not including the discussion in this thesis.

6.4.2 Boyd-Foo protocol

Now we discuss a non-effective protocol derived from the protocol given in [9]. The protocol uses the Gennaro-Krawczyk-Rabin (GKR) scheme [30] for designated-converter signature. A designated-verifier extension of the scheme is discussed in [30]. The designated-verifier and the designated-converter signature by O intended for R with the designated converter T is summarized as $S(pd, k_o, k_r, k_t)$. This is the realization of *vcsc* signature scheme discussed in chapter 2. Hence, $S(pd, k_o, k_r, k_t)$ can be converted into a universally verifiable signature. Although R and T can tell that this signature was indeed generated by O, R cannot use it convince an outside party that it was O who generated it (R can fake it).

This protocol differs from the GJM protocol in that only three messages are used in the exchange protocol. The exchange protocol starts with O by sending $S(pd, k_o, k_r, k_t)$ to R which verifies that it was generated by O using an interactive zero-knowledge proof. After verifying, R sends back $sig(k_r, pd)$ to O. Finally O sends back $sig(k_o, pd)$ to R. There is no abort subprotocol, and only R can ask T to resolve after it has sent $sig(k_r, pd)$. It does so by sending $sig(k_r, {<}S(pd, k_o, k_r, k_t),$ $sig(k_r, pd){>})$ over a read and write-protected channel. T can convert $S(pd, k_o, k_r, k_t)$ into a universally verifiable signature $sig(k_o, pd)$. T resolves by exchanging the signatures.

Exchange subprotocol.

$$\begin{array}{lrcl}
O \rightarrow R & me_1 & = & S(pd, k_o, k_r, k_t) \\
R \rightarrow O & me_2 & = & sig(k_r, pd) \\
O \rightarrow R & me_3 & = & sig(k_o, pd)
\end{array}$$

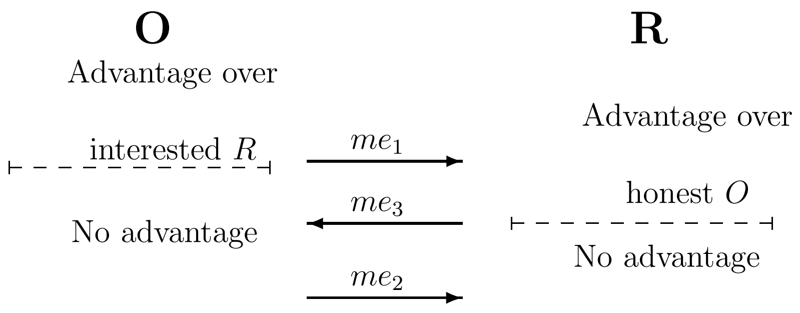


Figure 6.2: Advantage flow in Boyd-Foo protocol

Resolve subprotocol.

$$
\begin{array}{lll}
R \to T & ma_1 & = sig(k_r, <S(pd, k_o, k_r, k_t), sig(k_r, pd)>) \\
T \to R & mr_1 & = sig(k_o, pd). \\
T \to O & mr_2 & = sig(k_r, pd).
\end{array}
$$

The protocol can be shown to be fair but not effective for O. This is because R may not respond to me_1 and O will be left hanging. For the same reason the protocol is not balanced for honest O (and hence also for interested and optimistic O). To abort R never responds to me_1 and to complete the exchange, R contacts T. Once R sends me_1, nobody has an advantage. This indeed is the non-initial point that is captured in theorem 6.3.6. Figure 6.2 describes the advantage flow in this protocol:

Please note that in this protocol, only one signer enjoys an advantage against an optimistic opponent at a non-initial point.

Chapter 7

Conclusions and further work

We have studied in detail an optimistic two-party signature-exchange protocol derived from the Garay-Jakobsson-MacKenzie (GJM) protocol [28]. We state effectiveness and fairness for this protocol in the multiset-rewriting formalism [14, 13, 22]. Recall that a protocol is said to be effective if it provides means to the signers to prevent unbounded waiting, and a protocol is said to fair if it is the case that either both signers get each other's signatures or none does. Fairness and effectiveness are formally proved by inductive methods.

For this protocol, we also studied balance for honest signer's: at any stage of the protocol, any protocol signer does not have the power to unilaterally decide the outcome of the protocol. (It is a consequence of fairness that either both the signers will get each other's signature or none will). We use the multiset-rewriting formalism to formally state this property in terms of a certain recursive property of the protocol execution tree, which we then prove for our version of the GJM protocol by inductive methods.

We extended this model to study general two-party signature-exchange protocols in a game-theoretic model. We have given precise, formal definitions of desired security guarantees such as fairness and effectiveness in this model and characterized optimistic protocols using signals that allow an optimistic signer to wait for the other signer. We say that a protocol is optimistic for a signer, Alice, if it is the case that whenever Alice is willing to wait "long enough", then the other signer, Bob, has a strategy to exchange signatures without any of them contacting the third party.

Alice's willingness to wait long enough is modeled by giving Bob the ability to signal Alice the option to contact the third party. While these signals do not correspond to any global synchronized clocks, it accurately reduces the set of protocol traces to those where a biased signer waits for events that are forthcoming.

Using this model, we study bias of honest signers. We consider two kinds of biased signers: optimistic and interested. An optimistic signer is one who waits for some period of time before contacting the trusted third party, and an interested signer is one who waits for some period of time before quitting the exchange or contacting the trusted third party with a request to abort the exchange.

Our main result is that in any fair and optimistic protocol, some optimistic signer yields the other signer an advantage. In our terminology, a signer with an advantage has both a strategy to get the other signer's signature and a strategy to keep the other signer from getting one. By fairness, the outcome for both signers is the same, but the signer with an advantage can determine the outcome. We have applied our definitions to several signature-exchange protocols [3, 9, 28] and showed that these protocols are fair, effective and optimistic for single runs. In the dissertation, we show how advantage flows in the signature-exchange protocols given in [9, 28], and illustrate our result of impossibility of balance on these protocols.

Since advantage cannot be eliminated, the best a protocol can do to protect optimistic signers is to prevent the opponent from proving to any outside party that it has reached a position of advantage. Absence of probable advantage has been called abuse-freeness in the literature [3, 28].

One direction for further investigation is to study signature-exchange protocols when the third party misbehaves. In our analysis of two-party fair exchange protocols with a trusted third party, we assumed that the third party is always honest and never deviates from the protocol. There are some protocols, *e.g.* [27], which ensure that a limited misbehavior of the third party does not compromise certain security guarantees for honest signers. Certain other protocols [28] claim *trusted third party accountability* for their protocols. A fair exchange protocol is said to be third party accountable [28], if whenever an honest signer is cheated because the third party misbehaved, the honest signer is capable of proving to an arbitrator that the third party misbehaved. We would like to investigate formalizing possible dishonest behavior of the third party and formalizing security guarantees such as trusted third party accountability. Our current approach to modeling dishonesty of signers by giving them additional dishonest actions is a good match for modeling possible dishonest behavior of the third party. Another concept that we can borrow from our current work is the notion of "ability to prove", which we formalized using epistemic logic [33, 25].

In the dissertation, we are mainly concerned with two-party signature-exchange protocols and it would be interesting to investigate signature exchange protocols when the number of signers increases, *e.g.* [29]. A preliminary analysis of the protocol in [29] reveals some anomalies. Formal analysis of these multiparty protocols tends to be more involved. This is for several reasons. As a first level of complexity, there is more data to be handled. Each signer has to keep track of the status of its exchange with others and the trusted third party has to reconcile the differing status. In the case of [29] which is the extension of Garay, Jakobsson and Mackenzie protocol [28], the trusted third party may have to overturn its previous decisions to ensure fairness. Hence, database persistence which was one of the keys to proving fairness in the two-party version, is no longer valid in this protocol. As a second level of complexity, there can conceivably be coalitions amongst signers in order to gain an edge over other signers. The security guarantees may change with different coalitions. We would also like to investigate if our impossibility results on elimination of advantage lift in some form to multiparty protocols.

Automated proof search and finite state analysis has made security protocol analysis more tractable [43, 37]. We would like to investigate the use of these formal methods in this area. This should prove useful for example in to handle the complexity of multiparty protocols, Since we used multiset-rewriting formalism [14, 13, 22] for the study of two-party exchange protocols, which is closely related to linear logic [31], automated proof search in rewriting logics [39] and proof search in linear logic are natural candidates for analyzing such protocols. In particular, we would like to investigate using MAUDE for the analysis, which has been used for analysis of authentication protocols [20].

Bibliography

[1] N. Asokan, M. Schunter, and M. Waidner, *Optimistic protocols for fair exchange*, Proc. 4th ACM Conference on Computer and Communications Security (CCS-4), 1997, pp. 7–17.

[2] N. Asokan, V. Shoup, and M. Waidner, *Asynchronous protocols for optimistic fair exchange*, IEEE Symposium on Security and Privacy, 1998, pp. 86–99.

[3] ———, *Optimistic fair exchange of digital signatures*, IEEE Journal on Selected Areas in Communications **18** (2000), no. 4, 593–610.

[4] J. Banatre and D. Le Metayer, *Computing by multiset transformation*, Communications of the ACM (CACM) **36** (1993), no. 1, 98–111.

[5] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest, *A fair protocol for signing contracts*, IEEE Transactions on Information Theory **36** (1990), no. 1, 40–46.

[6] G. Berry and D. Boudol, *The chemical abstract machine*, Theoretical Computer Science **283** (2002), no. 2, 419–450.

[7] D. Boneh and M. Naor, *Timed commitments*, Advances in Cryptology – CRYPTO, vol. 1880, Springer- Verlag Lecture Notes in Computer Science, 2000, pp. 236–254.

[8] J. Boyar, D. Chaum, and I. B. Damågard, *Convertible undeniable signatures*, Advances in Cryptology - Proceedings of CRYPTO '90, 1990, pp. 189–205.

[9] C. Boyd and E. Foo, *Off-line fair payment protocols using convertible signatures*, Advances in Cryptology - Proceedings of ASIACRYPT '98, 1998, pp. 271–285.

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

[11] F. Butler, I. Cervesato, A. D. Jaggard, and A. Scedrov, *A Formal Analysis of Some Properties of Kerberos 5 Using MSR*, Fifteenth Computer Security Foundations Workshop (CSFW-15) (Cape Breton, NS, Canada), IEEE Computer Society Press, 24–26 June 2002, pp. 175–190.

[12] 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.

[13] I. Cervesato, N. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov, *A Comparison between Strand Spaces and Multiset Rewriting for Security Proto-*

col Analysis, Software Security - Theories and Systems — ISSS 2002 (Tokyo, Japan), Springer-Verlag LNCS 2609, 8–10 November 2003, pp. 356–383.

[14] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov, *A meta-notation for protocol analysis*, Proc. 12-th Annual IEEE Computer Security Foundations Workshop (CSFW-12) (Mordano, Italy), IEEE Computer Society Press, 1999, pp. 55–69.

[15] R. Chadha, M. Kanovich, and A. Scedrov, *Inductive methods and contract-signing protocols*, Proc. 8-th ACM Conference on Computer and Communications Security (CCS-8) (Philadelphia), ACM Press, 2001, pp. 176–185.

[16] R. Chadha, J. Mitchell, A. Scedrov, and V. Shmatikov, *Contract signing, optimism, and advantage*, To appear in Proc. 14th International Conference on Concurrency Theory (CONCUR '03), 2003.

[17] V. Cortier, J. Millen, and H. Ruess, *Proving secrecy is easy enough*, 14-th Annual IEEE Computer Security Foundations Workshop'01 (CSFW-14), 2001, pp. 97–108.

[18] I. B. Damågard, *Practical and provably secure release of a secret and exchange of signatures*, Journal of Cryptology **8** (1995), no. 4, 201–222.

[19] S. Das and D. Dill, *Successive approximation of abstract transition relations*, Sixteenth Annual IEEE Symposium on Logic in Computer Science, 2001, pp. 51–58.

[20] G. Denker, J. Meseguer, and C. Talcott, *Protocol specification and analysis in MAUDE*, Workshop on Formal Methods and Security Protocols, 1998.

[21] D. Dolev and A. Yao, *On the security of public-key protocols*, Proc. 22nd Annual IEEE Symp. Foundations of Computer Science, 1981, pp. 350–357.

[22] N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov, *Multiset rewriting and the complexity of bounded security protocols*, To appear in Journal of Computer Security.

[23] H. B. Enderton, *A mathematical introduction to logic*, 2 ed., Harcourt / Academic Press, 2000.

[24] S. Even and Y. Yacobi, *Relations among public key signature schemes*, Technical Report 175, Computer Science Dept. Technion, Israel, March 1980.

[25] R. Fagin, J. Halpern, Y. Moses, and M. Vardi, *Reasoning about knowledge*, MIT Press, 1995.

[26] M. Fischer, N. Lynch, and M. Patterson, *Impossibility of distributed consensus with one faulty process*, JACM **32** (1985), no. 2, 374–382.

[27] M. K. Franklin and M. K. Reiter, *Fair exchange with a semi-trusted third party (extended abstract)*, Proc. 4th ACM Conference on Computer and Communications Security (CCS-4), 1997, pp. 1–5.

[28] J. Garay, M. Jakobsson, and P. MacKenzie, *Abuse-free optimistic contract signing*, Advances in Cryptology – CRYPTO'99, Springer Lecture Notes in Computer Science, vol. 1666, 1999, pp. 449–466.

[29] J. A. Garay and P. D. MacKenzie, *Abuse-free multi-party contract signing*, International Symposium on Distributed Computing, 1999, pp. 151–165.

[30] R. Gennaro, T. Rabin, and H. Krawczyk, *RSA-based undeniable signatures*, Journal of Cryptology **13** (2000), no. 4, 397–416.

[31] J.-Y. Girard, *Linear logic*, Theoretical Computer Science **50** (1987), 1–102.

[32] N. Heintze, J. D. Tygar, J. M. Wing, and H.-C. Wong, *Model checking electronic commerce protocols*, Proc. USENIX 1996 Workshop on Electronic Commerce, 1996, pp. 147–164.

[33] J. Hintikka, *Knowledge and belief*, Cornell University Press, Ithaca, NY, 1962.

[34] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo, *Designated verifier proofs and their applications*, Advances in Cryptology - Proceedings of EUROCRYPT '96, 1996, pp. 143–154.

[35] S. Kremer and J.-F. Raskin, *A game-based verification of non-repudiation and fair exchange protocols*, Proc. 12th International Conference on Concurrency Theory (CONCUR '01), 2001, pp. 551–565.

[36] ———, *Game analysis of abuse-free contract signing*, Proc. 15th IEEE Computer Security Foundations Workshop (CSFW 15), 2002, pp. 206–220.

[37] Gavin Lowe, *Breaking and fixing the Needham-Schroeder public-key protocol using FDR*, Tools and Algorithms for the Construction and Analysis of Systems (TACAS), vol. 1055, Springer-Verlag, Berlin Germany, 1996, pp. 147–166.

[38] O. Markowitch and S. Saeednia, *Optimistic fair exchange with transparent signature recovery*, Proc. 5th International Conference on Financial Cryptography, 2001, pp. 339–350.

[39] J. Meseguer, *Conditioned rewriting logic as a unified model of concurrency*, Theoretical computer science **96** (1992), no. 1, 73–155.

[40] J.C. Mitchell and V. Shmatikov, *Analysis of abuse-free contract signing*, Financial Cryptography '00, 2000.

[41] R.M. Needham and M.D. Schroeder, *Using encryption for authentication in large networks of computers*, Communications of the ACM **21** (1978), no. 12, 993–999.

[42] 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.

[43] L. Paulson, *Proving properties of security protocols by induction*, Proc. 10th Computer Security Foundations Workshop, IEEE Computer Society Press, 1997, pp. 70–83.

[44] S. Schneider, *Formal analysis of a non-repudiation protocol*, Proc. 11th IEEE Computer Security Foundations Workshop, 1998, pp. 64–55.

[45] V. Shmatikov and J. C. Mitchell, *Finite-state analysis of two contract signing protocols*, Theoretical Computer Science **283** (2002), no. 2, 419–450.

[46] D. Stinson, *Cryptography: Theory and practice*, 2 ed., Chapman and Hall, 2002.

[47] T.Y.C. Woo and S.S. Lam, *A semantic model for authentication protocols*, Proc. IEEE Symposium on Research in Security and Privacy, 1993.

[48] J. Zhou and D. Gollmann, *A fair non-repudiation protocol*, Proc. IEEE Symposium on Security and Privacy, 1996, pp. 55–61.