GMD Report 150

GMD –
Forschungszentrum
Informationstechnik
GmbH

Sigrid Gürgens, Peter Ochsenschläger
Carsten Rudolph

Authenticity and Provability – a Formal Framework

2001

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

Anschrift der Verfasser/Address of the authors:
Dr. Sigrid Gürgens
Dr. Peter Ochsenschläger
Dr. Carsten Rudolph
Institut für sichere Telekooperation (SIT)
GMD – Forschungszentrum Informationstechnik GmbH
Rheinstraße 75
64204 Darmstadt
E-Mail: guergens@darmstadt.gmd.de
ochsenschlaeger@darmstadt.gmd.de
cartsen.rudolph@darmstadt.gmd.de

ISSN 1435-2702

Abstract

Authentication and non-repudiation are essential security requirements for electronic commerce applications and other types of binding telecooperation. Symmetric and asymmetric encryption techniques and different types of digital signatures can be used to provide these security services. However, cryptographic algorithms can only provide isolated functionality that has to be integrated into more or less complex cryptographic protocols. It is not always clear which security services the resulting protocol provides, making it hard to use the protocol appropriately. In this paper we present a formal method for the specification of e-commerce protocols and transactions on different levels of abstraction. Based on the notions of formal language theory we introduce formalisations of the security services of authenticity and proof of authenticity that are independent of the abstraction level. Language homomorphisms satisfying particular properties preserve the respective security properties from a higher to a lower level of abstraction.

Keywords: authenticity, provability, e-commerce, binding cooperations, formal languages, asynchronous product automata

Zusammenfassung

Authentizität und Nichtabstreitbarkeit sind essentielle Sicherheitsanforderungen von e-commerce und anderen verbindlichen Telekooperationsanwendungen. Symmetrische und asymmetrische Verschlüsselungstechniken und verschiedene Typen digitaler Signaturen können zur Realisierung dieser Sicherheitsanforderungen verwendet werden. Allerdings muss die durch kryptographische Algorithmen gelieferte isolierte Funktionalität in mehr oder weniger komplexe kryptographische Protokolle integriert werden. Es ist nicht immer klar ersichtlich, welche Sicherheitsanforderungen durch ein Protokoll erfüllt werden. Daher ist es oft schwierig, kryptographische Protokolle sicher einzusetzen. In diesem Bericht stellen wir eine formale Methode vor, mit der e-commerce Protokolle auf verschiedenen Abstraktionsstufen spezifiziert werden können. Basierend auf der Theorie formaler Sprachen werden die Sicherheitsdienste Authentizität und Nachweisbarkeit von Authentizität unabhängig vom jeweiligen Abstraktionsniveau formalisiert. Sprachhomomorphismen, die bestimmte Bedingungen erfüllen, erhalten die Eigenschaften beim Übergang von abstrakten zu konkreteren Modellen.

Schlagworte: Authentizität, Nachweisbarkeit, e-commerce, verbindliche Kooperationen, Formale Sprachen, Asynchrone Produktautomaten

Part of this work is based on results of a project funded by Siemens AG.

Contents

1 Introduction

Authentication and non-repudiation are essential security requirements for electronic commerce applications and other types of binding telecooperation. Symmetric and asymmetric encryption techniques and different types of digital signatures can be used to provide these security services. However, cryptographic algorithms can only provide isolated functionality that has to be integrated into more or less complex cryptographic protocols. Subtle differences between these protocols and the assumptions used make it hard to decide what is the security service and the related security property a protocol actually provides. This may result in a security flaw of the application, not because of a weakness of the protocol but because of a misinterpretation of the protocol's security properties.

Our approach for the systematic design of an e-commerce application is to specify the security services required for the application on a high level of abstraction independently of any implementations such as cryptographic protocols, and then to verify that a certain protocol provides the related security properties. This approach, formulated in terms of formal language theory, is based on a model for key establishment protocols [10] and is related to a formal framework for binding cooperations [5]. In that paper those states of a binding cooperation are determined in which proofs of authenticity are necessary to reach the goal of the cooperation.

In this paper, we present formal methods for the specification of e-commerce protocols, transactions and properties on different levels of abstraction. On a higher level of abstraction, the behaviour of a system is specified by identifying all possible actions and specifying the set of possible state transition sequences. On a lower abstraction level we use Asynchronous Product Automata (APA), the behaviour of which uniquely defines the set of possible state transition sequences [9]. Suitable language homomorphisms map from the lower to the higher level of abstraction. Independent of the abstraction levels, we introduce formal specifications of the security services authentication and proof of authentication in terms of state transition sequences, viewed from the agents' perspective. On the high level of abstraction, the respective security properties result in requirements on the system. These requirements correspond to security mechanisms on the lower abstraction level that are realized by using abstract communication channels that provide certain security properties. We finally define properties of language homomorphisms that preserve authenticity and proof of authenticity from a higher to a lower level of abstraction.

Most of the work in the field of cryptographic protocols has concentrated on the analysis of authentication and key establishment protocols. Therefore, formalisation of security properties is usually based on special representations of protocols on a fixed level of abstraction and restricted to authentication, freshness and confidentiality of keys. Our work is related to a range of work concerning general specification of security properties. Meadows and Syverson [13] define a formal language for the specification of requirements on cryptographic protocols. Authenticity is defined as the property that a message is only accepted if at some point in the past it was sent by a specific agent and before that requested from that agent. Schneider [11, 12] uses a more general approach to define authenticity of events or sets of events on globally viewed traces. Both approaches do not cover provability or non-repudiation of authenticated events towards other agents. In the context of anlysing electronic commerce protocols with respect to accountability, Kailar [6] introduces the notion of strong and weak proofs that can be transferable or nontransferable. He then uses inference rules to reason about accountability. Another framework to reason about properties of electronic commerce protocols was proposed by Clarke, Jha and Marrero [3].

However, both approaches do not provide the formalization of a proof itself. All these papers are exclusively concerned with the analysis of security protocols. To the best of our knowledge, there is no approach to a formalisation of security properties of electronic commerce protocols that supports a design methodology.

To illustrate our approach, we give the abstract specification of a simple e-commerce transaction and show that it provides authentication and proof of authentication. We then present a concrete protocol on the lower abstraction level where appropriate abstract channels are used. We finally define a homomorphism that maps the protocol specification to the abstract specification and show that it provides the necessary properties, thus proving that the protocol itself provides authenticity and proof of authenticity.

The rest of the paper is organized as follows. In the next section we present the formal definitions of the security services authentication and proof of authentication and we define homomorphism properties that preserve these security properties from a higher to a lower level of abstraction. In Section 3 we specify a simple price request and offer transaction on a high abstraction level and show that it provides authenticity and proof of authenticity. In Section 4 we introduce Asynchronous Product Automata and show how they are used to model protocols. In Section 5 we present a concrete protocol specification for the price request and offer transaction. We then specify a concrete homomorphism that maps the protocol onto the high level specification and show that it provides the desired properties. As we present an example on two specific levels of abstraction, we will refer to the higher level as to "the high level" and to the lower level as to "the low level" of abstraction.

2 Formal specification of security properties

2.1 System behaviour and abstraction by language homomorphisms

The behaviour S of a discrete system can be formally described by the set of its possible sequences of actions. Therefore $S \subseteq \Sigma^*$ holds where Σ is the set of all actions of the system and Σ^* the set of all finite sequences of elements of Σ, including the empty sequence denoted by ε. This terminology originates from the theory of formal languages [4], where Σ is called the alphabet, the elements of Σ are called letters, the elements of Σ^* are referred to as words and the subsets of Σ^* as formal languages. Words can be composed: if u and v are words, then uv is also a word. This operation is called the *concatenation*; especially $\varepsilon u = u\varepsilon = u$. A word u is called a *prefix* of a word v if there is a word x such that $v = ux$. The set of all prefixes of a word u is denoted by $\mathrm{pre}(u)$; $\varepsilon \in \mathrm{pre}(u)$ holds for every word u. We denote the set of letters in a word u by $alph(u)$.

Formal languages which describe system behaviour have the characteristic that $\mathrm{pre}(u) \subseteq S$ holds for every word $u \in S$. Such languages are called *prefix closed*. System behaviour is thus described by prefix closed formal languages.

The set of all possible continuations of a word $u \in S$ is formally expressed by the *left quotient* $u^{-1}(S) = \{y \in \Sigma^* \mid uy \in S\}$.

Different formal models of the same application/system are partially ordered with respect to different levels of abstraction. Formally, abstractions are described by so called alphabetic language homomorphisms. These are mappings $h^* : \Sigma^* \longrightarrow \Sigma'^*$ with $h^*(xy) = h^*(x)h^*(y)$, $h^*(\varepsilon) = \varepsilon$ and $h^*(\Sigma) \subseteq \Sigma' \cup \{\varepsilon\}$. So they are uniquely defined by corresponding mappings $h : \Sigma \longrightarrow \Sigma' \cup \{\varepsilon\}$. In the following we denote both the mapping h and the homomorphism h^* by h. These homomorphisms map action sequences of a finer abstraction level to action sequences of a more abstract level.

Consider for example an application consisting of four actions: a price for a certain service is requested, the request is received, and then an offer for this service is sent and received. On a high level of abstraction this may be formalized by only specifying the names of the actions (e.g. PRICEREQ-S, PRICEREQ-R, OFFER-S, OFFER-R) and system assumptions such as an offer can only be received if at some point before it was sent. On a lower level of abstraction more information about the system may be specified: The agents' actions may be split into internal and external ones, their internal memory may be specified, etc. A homomorphism can be defined that maps the internal actions of the agents onto ε, all send actions of price requests onto PRICEREQ-S, all price request receive actions onto PRICEREQ-R, etc. This homomorphism serves as an abstraction of the finer system specification.

2.2 Malicious behaviour

Usually, the specification of a cryptographic protocol does not include possible malicious behaviour of external attackers or of malicious protocol agents. In order to be able to express security properties, malicious behaviour has to be considered. For the security analysis of protocols malicious behaviour is often explicitly specified and included in the system behaviour. However, in general malicious behaviour is not previously known and one may not be able to adequately specify all possible actions of dishonest agents. Therefore, in this paper we use a different approach. Malicious behaviour is not explicitly specified but restricted by system assumptions and by assumptions about the underlying security mechanisms (for example the application of cryptography).

Let Σ be any set of actions (that may contain malicious actions) and $S_C \subseteq \Sigma^*$ a correct system behaviour without malicious actions. A behaviour containing malicious action is denoted by S. We assume that $S_C \subseteq S \subseteq \Sigma^*$. Let $\mathbb{P}$ be a set of agents. For each $P \in \mathbb{P}$ we denote by $W_P \subseteq \Sigma^*$ the set of those sequences agent P considers to be possible. W_P formalizes P's knowledge about a system S_C. The set Σ^* as well as the sets W_P may contain malicious behaviour. Both sets are not completely specified and are in general infinite, but all sets W_P are restricted by the following assumptions:

- W_P satisfies properties of underlying security mechanisms (see Section 4.4 for formal definitions of security mechanisms of a system S).
- System assumptions cannot be violated in W_P. (For example the assumption that access to an agent's internal memory is protected as provided by the structure of the model used in Section 5).

We assume $S_C \subseteq W_P$, i.e. every agent considers the correct system behaviour to be possible. Security properties can now be defined relative to W_P. A system S may satisfy a security property with respect to W_P, but may fail to satisfy the same property with respect to a different $\tilde{W}_P$, if P considers more actions to be possible on account of weaker system assumptions and security mechanisms.

After a sequence of actions $\omega \in S$ has happened, every agent can only use his *local view* of ω to determine the sequences of actions he considers to be possible. In order to determine what is the local view of an agent, we first observe that the actions Σ can be seperated into those that are relevant for a security property and those that are not, the latter set of actions being denoted by $\Sigma_{/\#}$. In the above introduced price request/offer example, we may want the offer to provide certain security properties, while the sending and receiving of

a price-request are not relevant for any security property and therefore belong to $\Sigma_{/\#}$. All actions relevant for a security property (sending and receiving of the offer) are performed by and can be assigned to exactly one agent. Thus $\Sigma = \dot{\bigcup}_{P \in \mathbb{P} \cup \{\#\}} \Sigma_{/P}$ (where $\Sigma_{/P}$ denotes all actions performed by agent P). The homomorphism $\pi_P : \Sigma^* \to \Sigma^*{}_{/P}$ defined by $\pi_P(x) = x$ if $x \in \Sigma_{/P}$ and $\pi_P(x) = \varepsilon$ if $x \in \Sigma \setminus \Sigma_{/P}$ formalizes the assignment of actions to agents and is called the *projection* on P.

On a high level of abstraction where the system behaviour is described by sequences of actions containing essentially the action name and the agent performing the action (if any), all information about an action $x \in \Sigma_{/P}$ is available for agent P. Thus π_P defines P's local view of the system on this level of abstraction. But on a lower level of abstraction, where additional information such as the agents' memory may be available, the elements of Σ may contain information about the global system state (e.g. all agents' memory) and may be represented by a triple (global state, transition lable, global successor state). However, an agent P generally cannot "see" the complete global state (he cannot see, for example, other agents' memory). Therefore, the projection π_P may be too fine to define the local view of an agent $P \in \mathbb{P}$. Thus, we generally denote the local view of an agent P on Σ by $\lambda_P : \Sigma^* \to \Sigma_P^*$. The local views of all agents together with the behaviour not assigned to any agent contain all information about the system behaviour S. For every abstraction level, the local views of the agents have to be defined separately and will in general be different.

For a sequence of actions $\omega \in S$ and agent $P \in \mathbb{P}$, $\lambda_P^{-1}(\lambda_P(\omega)) \subseteq \Sigma^*$ is the set of all sequences that look exactly the same from P's local view after ω has happened. But depending on his knowledge about the system S and underlying security mechanisms and system assumptions, P does not consider all sequences in this set possible. Thus he can use his knowledge to reduce this set: $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$ describes all sequences of actions P considers to be possible when ω has happened.

2.3 Authenticity and proof of authenticity

Authenticity and proof of authenticity are important security properties for e-commerce applications and may also be relevant for the price-request/offer example. The agent receiving an offer may want to know who sent it, and may also want to be able to proof that the offer was sent by a certain other agent.

In this section we formally define the security properties authenticity and proof of authenticity and then give sufficient conditions on language homomorphism to preserve these properties from higher to lower levels of abstraction.

We define authenticity from the agents' perspective. A set of actions Γ is authentic for agent P if in all sequences that P consideres possible after a sequence of actions has happened, some time in the past an action in Γ must have happened. One (or more) of the actions of ω performed by P is responsible for Γ being authentic for P. This can be, for example, that P receives an offer. If then all sequences of actions he considers possible contain a send offer action performed by Q, then the set of actions where Q sends an offer is authentic for P.

In the following let $S \subseteq \Sigma^*$ be a prefix closed language describing the behaviour of a system and $\mathbb{P}$ a set of agents.

Definition 1 (Authenticity) *A set of actions $\Gamma \subseteq \Sigma$ is authentic for $P \in \mathbb{P}$ after a sequence of actions $\omega \in S$ with respect to W_P if $alph(x) \cap \Gamma \neq \emptyset$ for all $x \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$.*

Some actions do not only require authenticity but also need to provide a proof of authenticity. If agent P owns a proof of authenticity for a set Γ of actions, he can send this proof to other agents, which in turn can receive the proof. In the following definition the set $\Gamma\mathrm{P}$ denotes actions that provide agents with proofs about the authenticity of Γ. If agent P has executed an action from $\Gamma\mathrm{P}$ then Γ is authentic for P and P can forward the proof to any other agent using actions in $\Gamma\mathrm{S}$.

Definition 2 (Proof of authenticity) *A pair* $(\Gamma\mathrm{S}, \Gamma\mathrm{P})$ *with* $\Gamma\mathrm{S} \subseteq \Sigma$ *and* $\Gamma\mathrm{P} \subseteq \Sigma$ *is a pair of sets of proof actions of authenticity for a set* $\Gamma \subseteq \Sigma$ *on* S *with respect to* $(W_P)_{P\in\mathbb{P}}$ *if for all* $\omega \in S$ *and for all* $P \in \mathbb{P}$ *with* $alph(\pi_P(\omega)) \cap \Gamma\mathrm{P} \neq \emptyset$ *the following holds:*

1. *For* P *the set* Γ *is authentic after* ω *and*
2. *for each* $R \in \mathbb{P}$ *there exists actions* $a \in \Sigma_{/P} \cap \Gamma\mathrm{S}$ *and* $b \in \Sigma_{/R} \cap \Gamma\mathrm{P}$ *with* $\omega ab \in S$.

Agent $P \in \mathbb{P}$ *can give proof of authenticity of* $\Gamma \subseteq \Sigma$ *after a sequence of actions* $\omega \in S$ *if 1 and 2 hold.*

In the following, we shortly call $(\Gamma\mathrm{S}, \Gamma\mathrm{P})$ a *proof action pair for* Γ.

This definition represents one specific type of proofs. Kailar has classified proofs as strong or weak and transferable or non-transferable [6]. In terms of this classification our definition provides strong transferable proofs with the additional property that the possibility of the proof transfer is reliable. These proofs require the assumption that no agent disposes of his proofs. From a technical point of view the formal definition of this property is the most simple. However, other types of proofs can be formalized in a similar way. For example by introducing an additional class of actions representing the loss of proofs, Definition 2 can be modified: Agent P can give proof of authenticity only as long as no corresponding loss action has occured. The appendix contains corresponding modifications of Definition 2, Definition 4, and Theorem 2.

As already explained, we describe an application on different levels of abstraction. On a lower level we may describe the system behaviour by using tupels (global state, action, global successor state) that contain information about agents' memory etc. These tupels can easily be mapped to the respective actions of the higher abstraction level by use of an appropriate homomorphism. However, properties that hold on the high abstraction level need not necessarily hold on the low level and vice versa. The following two definitions give sufficient conditions of homomorphisms such that these "transport" certain security properties from a high to a low abstraction level.

Definition 3 *Let* $h : \Sigma^* \to \Sigma'^*$ *be an alphabetic homomorphism and for* $P \in \mathbb{P}$ *let* $\lambda_P : \Sigma^* \to \Sigma_P^*$ *and* $\lambda'_P : \Sigma'^* \to \Sigma_P'^*$ *be the homomorphisms describing the* local views *of* P *on* Σ *and* Σ'*, respectively. The language homomorphism* h *preserves authenticity on* S *if for each* $P \in \mathbb{P}$ *exists a mapping* $h'_P : \lambda_P(S) \to \lambda'_P(S')$ *with* $\lambda'_P \circ h = h'_P \circ \lambda_P$ *on* S.

$f \circ g$ denotes the composition of functions f and g. The above property captures the fact that the homomorphism h has to be consistent with the local views of P on both abstraction levels in order to preserve authenticity. This means that the actions relevant for Γ' being authentic for P and their inverse images under h must be equally visible by P. In particular, the inverse image of the action being responsible for the authenticity of Γ' must not be mapped to ε by P's local view λ_P on the lower abstraction level.

Theorem 1 *If $\Gamma' \subseteq \Sigma'$ is authentic for $P \in \mathbb{P}$ after $\omega' \in h(S)$ with respect to W'_P, and if h preserves authenticity on S, then $\Gamma = h^{-1}(\Gamma') \cap \Sigma$ is authentic for $P \in \mathbb{P}$ after each $\omega \in h^{-1}(\omega') \cap S$ with respect to each W_P with $W_P \subseteq h^{-1}(W'_P)$.*

Proof: Let $x \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$. Then $\lambda_P(x) = \lambda_P(\omega)$ and $x \in W_P$. Hence $h'_P(\lambda_P(x)) = h'_P(\lambda_P(\omega))$, which implies $\lambda'_P(h(x)) = \lambda'_P(h(\omega))$ by Definition 3. Hence $h(x) \in \lambda'^{-1}_P(\lambda'_P(h(\omega)))$. Since by assumption $W_P \subseteq h^{-1}(W'_P)$ and $x \in W_P$, $h(x) \in W'_P$ and therefore $h(x) \in \lambda'^{-1}_P(\lambda'_P(h(\omega))) \cap W'_P$. Now, by the definition of authenticity of Γ', $alph(h(x)) \cap \Gamma' \neq \emptyset$, which implies $alph(x) \cap \Gamma \neq \emptyset$. □

We now define properties for a homomorphism in order to preserve proofs. The first condition essentially states that whenever the homomorphic image of a sequence of actions $u \in S$ can in the abstract system be continued with a proof send action in $\Gamma\mathrm{S}'$ and a proof receive action in $\Gamma\mathrm{P}'$, there must be appropriate proof send and receive actions in the inverse image of $\Gamma\mathrm{S}'$ and $\Gamma\mathrm{P}'$, respectively, to continue the sequence of actions u in S.

The second condition states that h must not map actions performed by and assigned to agent P on the low abstraction level onto actions assigned to a different agent on the high abstraction level.

Note that $((\Gamma\mathrm{S}' \cap \Sigma'_{/P})(\Gamma\mathrm{P}' \cap \Sigma'_{/R}))$ is the set of all words of length 2 with the first letter in $(\Gamma\mathrm{S}' \cap \Sigma'_{/P})$ and the second letter in $(\Gamma\mathrm{P}' \cap \Sigma'_{/R})$.

Definition 4 *Let $\Gamma\mathrm{S}' \subseteq \Sigma'$ and $\Gamma\mathrm{P}' \subseteq \Sigma'$. $h : \Sigma^* \to \Sigma'^*$ preserves $(\Gamma\mathrm{S}', \Gamma\mathrm{P}')$-proofs on S if h preserves authenticity on S and if the following holds:*

1. *$h(u)^{-1}(h(S)) \cap ((\Gamma\mathrm{S}' \cap \Sigma'_{/P})(\Gamma\mathrm{P}' \cap \Sigma'_{/R})) \neq \emptyset$ implies $u^{-1}(S) \cap ((h^{-1}(\Gamma\mathrm{S}') \cap \Sigma_{/P})(h^{-1}(\Gamma\mathrm{P}' \setminus \Sigma'_{/\#}) \cap \Sigma_{/R})) \neq \emptyset$ for each $P, R \in \mathbb{P}$ and $u \in S$.*
2. *$h(\Sigma_{/P}) \cap (\Sigma' \setminus \Sigma'_{/\#}) \subseteq \Sigma'_{/P}$ for each $P \in \mathbb{P}$ and $h(\Sigma_{/\#}) \subseteq \Sigma'_{/\#}$.*

Theorem 2 *Let $(\Gamma\mathrm{S}', \Gamma\mathrm{P}')$ be a proof action pair for $\Gamma' \subseteq \Sigma'$ on $h(S)$ with respect to $(W'_P)_{P \in \mathbb{P}}$. Let $h : \Sigma^* \longrightarrow (\Sigma')^*$ be a homomorphism that preserves $(\Gamma\mathrm{S}', \Gamma\mathrm{P}')$-proofs on S, and let $\Gamma\mathrm{S} = h^{-1}(\Gamma\mathrm{S}') \cap \Sigma$, $\Gamma\mathrm{P} = h^{-1}(\Gamma\mathrm{P}' \setminus \Sigma'_{/\#}) \cap \Sigma$ and $\Gamma = h^{-1}(\Gamma') \cap \Sigma$. Then:*

1. *$(\Gamma\mathrm{S}, \Gamma\mathrm{P})$ is a proof action pair for $\Gamma \subseteq \Sigma$ on S ,*
2. *if agent $P \in \mathbb{P}$ can give proof of authenticity of Γ' after $\omega' \in h(S)$ then P can give proof of authenticity of Γ after each $\omega \in h^{-1}(\omega') \cap S$*

with respect to each family $(W_P)_{P \in \mathbb{P}}$ satisfying $W_P \subseteq h^{-1}(W'_P)$ for all $P \in \mathbb{P}$.

Proof: If $\omega \in S$ and $alph(\pi_P(\omega)) \cap \Gamma\mathrm{P} \neq \emptyset$ then $h(\omega) \in h(S)$ and $alph(\pi'_P(h(\omega))) \cap \Gamma\mathrm{P}' \neq \emptyset$ by the second condition of Definition 4. Now authenticity of Γ for P after ω follows from authenticity of Γ' for P after $h(\omega)$ by Theorem 1. By assumption, for each $R \in \mathbb{P}$ exist actions $a' \in \Sigma'_{/P} \cap \Gamma\mathrm{S}'$ and $b' \in \Sigma'_{/R} \cap \Gamma\mathrm{P}'$ with $h(\omega)a'b' \in h(S)$. This implies $h(\omega)^{-1}(h(S)) \cap ((\Gamma\mathrm{S}' \cap \Sigma'_{/P})(\Gamma\mathrm{P}' \cap \Sigma'_{/R})) \neq \emptyset$. Hence by Definition 4 there exist actions $a \in \Gamma\mathrm{S} \cap \Sigma_{/P}$ and $b \in \Gamma\mathrm{P} \cap \Sigma_{/R}$ with $\omega ab \in S$. This completes the proof of 1. Proposition 2 is shown similarily. □

It is easy to show that the property of preserving authenticity and proof of authenticity is closed under composition, i.e. if homomorphisms $h : \Sigma^* \longrightarrow (\Sigma')^*$ and $h' : (\Sigma')^* \longrightarrow (\Sigma'')^*$ preserve authenticity and proof of authenticity, so does $h' \circ h : \Sigma^* \longrightarrow (\Sigma'')^*$.

In some cases it may be more convenient to prove security properties on the low abstraction level and to use a suitable homomorphism to "transport" them to the higher level. We have defined sufficient conditions for both preserving authenticity and proof of authenticity with respect to abstraction. We omit these conditions in this paper because for the example to be introduced in the next section we can easily proof that both security properties hold on the abstract level.

3 An example

3.1 Abstract specification of a price offer

In this section the security properties defined above are demonstrated in the first step of a typical e-commerce situation, namely the price offer/request protocol already introduced in Section 2.1. As sending and receiving of the price request may not be relevant for the security of the transaction, we only consider the situation in which service providers make offers to clients. We assume that every agent $P \in \mathbb{P}$ can act as client and receive offers, while service providers are in the set $\mathbb{SP} \subseteq \mathbb{P}$. For simplicity we do not explicitly specify prices and services. We write PRO to denote the price offer example. Since we specify the system on a high abstraction level, alphabet, system, etc., will be denoted by Σ', S'_{PRO}, etc., in contrast to alphabet Σ, system S_{PRO} etc. on the low level abstraction level.

To formalize on a most abstract level the security property that each client can prove the authenticity of an offer received, we have to consider the following actions:

(1) service provider SP makes an offer: OFFER-$\mathrm{S_{SP}}$,

(2) client P receives a proof of (1): PROOF-$\mathrm{R_P}(SP)$,

(3) client P sends a proof of (1): PROOF-$\mathrm{S_P}(SP)$.

On this abstract level we make no distinction between receiving an offer from SP and receiving a proof that SP made an offer. These three types of actions exactly fit to the sets $\Gamma, \Gamma\mathrm{P}, \Gamma\mathrm{S}$ defined in the previous chapter.
So the action set of our abstract system is given by

$$\Sigma' = \bigcup_{SP\in\mathbb{SP},P\in\mathbb{P}} \{\text{OFFER-S}_{\mathrm{SP}},\ \text{PROOF-R}_{\mathrm{P}}(SP), \text{PROOF-S}_{\mathrm{P}}(SP)\}.$$

Now, in our framework we have to consider each agent's knowledge W'_P about the possible sequences of actions. To formalize proof of authenticity of offers, each agent has to know that a proof can only be received if a corresponding offer has been sent before, and that a proof can only be sent if it has been received before.

So for $P \in \mathbb{P}$ let

$$W'_P = \Sigma'^* \setminus \Big(\bigcup_{SP\in\mathbb{SP},Q\in\mathbb{P}} \begin{array}{l} (\Sigma' \setminus \{\text{OFFER-S}_{\mathrm{SP}}\})^*\{\text{PROOF-R}_{\mathrm{Q}}(SP)\}\Sigma'^* \cup \\ (\Sigma' \setminus \{\text{PROOF-R}_{\mathrm{Q}}\})^*\{\text{PROOF-S}_{\mathrm{Q}}(SP)\}\Sigma'^* \end{array} \Big)$$

By this definiton all W'_P are equal and we consider the abstract system behaviour $S'_{PRO} = W'_P$ for each $P \in \mathbb{P}$.

As mentioned in Section 2.2, at this level of abstraction we can define an agent's local view by $\lambda'_P = \pi'_P$ using the following definition:

For $P \in \mathbb{P}$ let

$$\Sigma_{/P} = \begin{cases} \bigcup_{SP \in \mathbb{S}} & \{\text{OFFER-S}_\text{P}, \text{PROOF-R}_\text{P}(SP), \text{PROOF-S}_\text{P}(SP)\} \quad if\ P \in \mathbb{SP} \\ \bigcup_{SP \in \mathbb{S}} & \{\text{PROOF-R}_\text{P}(SP), \text{PROOF-S}_\text{P}(SP)\} \quad if\ P \in \mathbb{P} \setminus \mathbb{SP} \end{cases}$$

The system S'_{PRO} shall satisfy the following properties:

1. An offer has to be authentic for an agent P receiving a corresponding proof and
2. the offer has to be obligatory, i.e. the agent has to be able to give proof of authenticity of the offer.

In accordance with Definition 2 the properties of authenticity and proof of authenticity of the offer can be specified as follows:

Property 1 *For each* $SP \in \mathbb{SP}$, $(\{\text{PROOF-S}_\text{P}(SP) | P \in \mathbb{P}\}, \{\text{PROOF-R}_\text{P}(SP) | P \in \mathbb{P}\})$ *is a proof action pair of authenticity for* $\{\text{OFFER-S}_\text{SP}\}$ *with respect to* $(W'_P)_{P \in \mathbb{P}}$.

Proof:

Condition 1 Let $\omega \in S'_{PRO}$ with $alph(\pi'_P(\omega)) \cap \{\text{PROOF-R}_\text{P}(SP) | P \in \mathbb{P}\} \neq \emptyset$. Hence we have $\text{PROOF-R}_\text{P}(SP) \in alph(\omega)$ which implies $\text{PROOF-R}_\text{P}(SP) \in alph(x)$ for each $x \in \lambda'^{-1}_P(\lambda'_P(\omega))$. By the definition of W'_P this implies $\text{OFFER-S}_\text{P}(SP) \in alph(x)$ for each $x \in \lambda'^{-1}_P(\lambda'_P(\omega)) \cap W'_P$.

Condition 2 In particular, for $x = \omega$, $\text{PROOF-R}_\text{P}(SP) \in alph(\omega)$ and $\text{OFFER-S}_\text{P}(SP) \in alph(\omega)$. Hence $\omega\text{PROOF-S}_\text{P}(SP)\text{PROOF-R}_\text{R}(SP) \in S'_{PRO}$ for each $R \in \mathbb{P}$ because none of these action sequences violates the restrictions defining S'_{PRO}. □

4 Asynchronous product automata

4.1 The formal definition

On the lower abstraction level, we model a system of protocol agents using asynchronous product automata (APA). APA are a universal and very flexible operational description concept for cooperating systems [9]. It "naturally" emerges from formal language theory [8]. APA are supported by the SH-verification tool that provides components for the complete cycle from formal specification to exhaustive validation [9].

An APA can be seen as a family of elementary automata. The set of all possible states of the whole APA is structured as a product set; each state is divided into state components. In the following the set of all possible states is called state set. The state sets of elementary automata consist of components of the state set of the APA. Different elementary automata are "glued" by shared components of their state sets. Elementary automata can "communicate" by changing shared state components.

Figure 1 shows a graphical representation of a small asynchronous product automaton consisting of two elementary automata e1 and e2 and state components C1 and C2, with state sets Z_{C1} and Z_{C2}. The state set of the APA as well as the state set of e1 is the product of Z_{C1} and Z_{C2}. The state set of e2 is Z_{C2}. The figure shows the structure of the automaton. The circles represent state components and a box corresponds to one elementary automaton. The full specification of the automaton includes the transition relations of the elementary automata and the initial state. The neighbourhood relation N indicates which

state components are included in the state of an elementary automaton and may be changed by a state transition of the elementary automaton. A state transition of automaton e1 may change the content of C1 and C2 while e2 may only change C2.

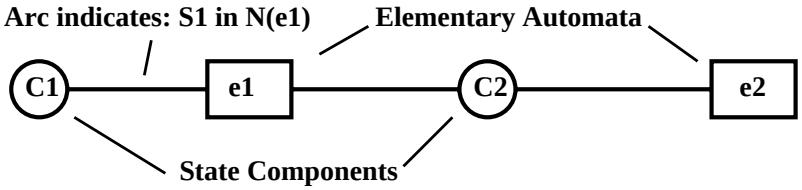

Figure 1: Graphical representation of an APA with two elementary automata e1 and e2 and state components C1 and C2

An *Asynchronous Product Automaton* consists of a family of *State Sets* $Z_S, S \in \mathbb{S}$, a family of *Elementary Automata* $(\Phi_e, \Delta_e), e \in \mathbb{E}$ and a *Neighbourhood Relation* $\mathrm{N} : \mathbb{E} \to \mathcal{P}(\mathbb{S})$; $\mathcal{P}(X)$ is the power set of X and $\mathbb{S}$ and $\mathbb{E}$ are index sets with the names of state components and elementary automata. For each Elementary Automaton (Φ_e, Δ_e)

- Φ_e is its *Alphabet* and
- $\Delta_e \subseteq \bigtimes_{S \in N(e)}(Z_S) \times \Phi_e \times \bigtimes_{S \in N(e)}(Z_S)$ is its *State Transition Relation*

For each element of Φ_e the state transition relation Δ_e defines state transitions that change only the state components in $N(e)$.

Remark: The alphabets Φ_e of the elementary automata can be used in protocol specifications to assign a label to any set of state transitions representing a particular protocol step (see below).

An APA's *States* are elements of $\bigtimes_{S \in \mathbb{S}}(Z_S)$. To avoid pathological cases it is generally assumed that $\mathbb{S} = \bigcup_{e \in \mathbb{E}}(N(e))$ and $N(e) \neq \emptyset$ for all $e \in \mathbb{E}$. Each APA has one *Initial State* $s_0 = (q_{0S})_{S \in \mathbb{S}} \in \bigtimes_{S \in \mathbb{S}}(Z_S)$.
In total, an APA $\mathbb{A}$ is defined by $\mathbb{A} = ((Z_S)_{S \in \mathbb{S}}, (\Phi_e, \Delta_e)_{e \in \mathbb{E}}, N, s_0)$.

The behaviour of an APA is represented by all possible sequences of state transitions starting with initial state s_0. The sequence $(s_0, (e_1, a_1), s_1)(s_1, (e_2, a_2), s_2)(s_2, (e_2, a_3), s_3) \ldots$ with $a_i \in \Phi_{e_i}$ represents one possible sequence of actions of the APA in Figure 1. In the following, in a sequence of state transitions ω, the successor state of state s is denoted by $\bar{s}$, state s_i occurring before state s_j is denoted by $s_i < s_j$. A state transition sequence ω ending in state s is denoted by ω_s.

4.2 APA specifications

For the formal specification of an APA, we have to specify all its components.

Elementary automata, state components and neighbourhood relationship $\mathbb{E}$, $\mathbb{S}$ and N can be defined using a graphical representation of the APA as shown in Figure 1.

State sets For each state component $C \in \mathbb{S}$ its state set Z_C has to be defined. The state of a state component is a multiset. Therefore, state sets are sets of multisets. A multiset of a set M is formally defined as a function $f \in \mathbb{N}^M$, where for $x \in M$, $f(x)$ indicates the

multiplicity of x in the multiset of M. If $f(x) > 0$ for $x \in M$, we say that x is element of the multiset f of M (denoted by $x \in f$). $\mathbb{N}^M$ is the set of all multisets of M. For each state component $C \in \mathbb{S}$ a set M_C has to be specified such that the state set Z_C is defined as $Z_C = \mathbb{N}^{M_C}$.

State transition pattern notation for APAs For the definition of the state transition relation of an elementary automaton $e \in \mathbb{E}$, we need to specify the properties of all states of components $C \in N(e)$ where e is active, i.e. can perform a state transition, and the changes of the states caused by the state transition. State transitions are defined by so-called *state transition patterns* in the following way:

a $(x_1, \ldots, x_n)$

$term_1 \in C_1(s) \wedge \ldots \wedge term_k \in C_k(s)$
allocations
predicates
$\xrightarrow{\mathbf{e}}$
$term_h \hookleftarrow C_h(s) \wedge \ldots \wedge term_j \hookrightarrow C_j(s)$

where a is the name of the transition pattern and $(x_1, \ldots, x_n)$ are the variables that occur in the pattern. $C_r \in N(e)$ are state components that can be accessed by e, $C_r(s)$ denotes one element of Z_{C_r} describing the state of C_r in the global state s, and $term_1, \ldots, term_k$ are terms in which the variables $(x_1, \ldots, x_n)$. $\xrightarrow{\mathbf{e}}$ indicates that the state transitions a are performed by automaton e. The lines above the arrow $\xrightarrow{\mathbf{e}}$ describe conditions that have to be satisfied by state s for the state transition to occur. Seperate lines are combined with $\wedge$. The expression $term_h \hookleftarrow C_h(s)$ denotes that $C_h(\overline{s})(term_h) = C_h(s)(term_h) - 1$ (i.e. the multiplicity of $term_h$ in $C_h(s)$ is decremented by one). $term_h \hookleftarrow C_h(s)$ is only defined if $term_h \in C_h(s)$. The expression $term_j \hookrightarrow C_j(s)$ denotes that $C_j(\overline{s})(term_j) = C_j(s)(term_j) + 1$ (i.e. the multiplicity of $term_j$ in $C_j(s)$ is incremented by one). $term_j \hookrightarrow C_j(s)$ is only defined if $term_j \in M_{C_j}$.

Formally, $(e, a, (x_1 = i_1, \ldots, x_n = i_n))$ is an element of the alphabet of e. It describes possible state transitions, i.e. tripels $((C_1(s), \ldots, C_k(s)), (e, a, (x_1 = i_1, \ldots, x_n = i_n), (C_1(\overline{s}), \ldots, C_k(\overline{s}))) \in \Delta_e$, where $x_1 = i_1, \ldots, x_n = i_n$ are the respective interpretations of the variables of the state transition pattern. For $(e, a, (x_1 = i_1, \ldots, x_n = i_n)) \in \Phi(e)$, let $pre(a, term_i, C_r(s))$ be the number of expressions $term_i \hookleftarrow C_r(s)$ and let $post(a, term_i, C_r(s))$ be the number of expressions $term_i \hookrightarrow C_r(s)$. A state transition can occur when an interpretation of the terms $term_1, \ldots, term_k$ exists such that

1. all conditions are satisfied and all expressions are defined, and
2. $pre(a, term_i, C_r(s)) \leq C_r(s)(term_i)$ for all terms $term_i$

Thus $(C_1(\overline{s}), \ldots, C_k(\overline{s}))$ describe the successor state by
$C_r(\overline{s})(term_i) = C_r(s)(term_i) - pre(a, term_i, C_r(s)) + post(a, term_i, C_r(s))$.

Hence the statements before $\xrightarrow{\mathbf{e}}$ constitute the prerequisites of e to perform a state transition and the statements behind $\xrightarrow{\mathbf{e}}$ describe the changes of the state. Elementary automaton e does not perform any additional changes to any state component's states during this state transition.

4.3 APA model for protocols

Each protocol participant P is modelled by four elementary automata $\mathrm{Send_P}, \mathrm{Rec_P}, \mathrm{App_P}$ and $\mathrm{Internal_P}$ and four state components $\mathrm{Channel_P}$, $\mathrm{Internal_P}$,$\mathrm{State_P}$ and $\mathrm{ApplMem_P}$. There is a further state component *Network* that is used for the communication between the agents. Figure 2 shows a graphical representation of the APA for a system of two protocol agents A and B.

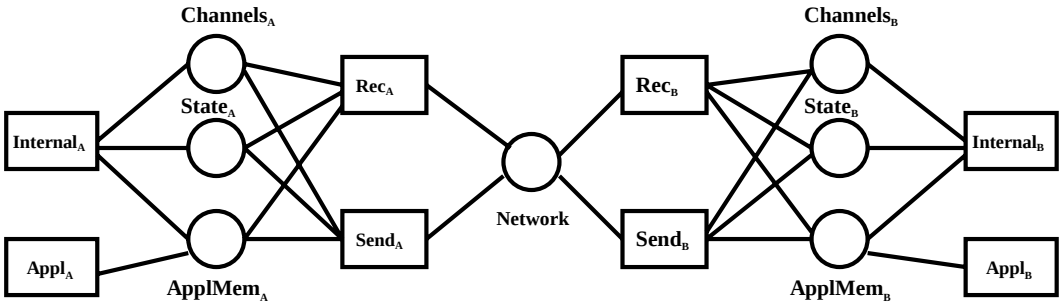

Figure 2: APA model with two protocol agents A and B

No elementary automaton of an agent P has access to a different agent's state components. The only state component shared between all agents is the component *Network*, which is used for communication. The general idea of how to use such an APA for modelling a protocol is the following: The automaton $\mathrm{Appl_P}$ serves as the interface between protocol and application. It specifies how applications request security services and access the result, i.e. it adds tupels to $\mathrm{ApplMem_P}$ in order to direct the actions of the protocol and removes tupels which represent the results of some protocol steps. The automata $\mathrm{Send_P}$ and $\mathrm{Rec_P}$ perform the communication between different agents (they insert into and remove tupels from *Network*, respectively). Messages are marked with a tag indicating the particular abstract channel they are sent on (see below). $\mathrm{Internal_P}$ is used for P's internal actions (such as random number generation, although not needed for the example in this paper). The state component $\mathrm{State_P}$ serves as the agent's internal memory which can be used to perform checks during the run of the protocol and $\mathrm{Channel_P}$ holds the channels P is able to send on and to receive from, respectively.

4.4 Abstract secure channels

Security requirements on e-commerce protocols are usually realised using cryptographic algorithms. In contrast to computational models for cryptography where properties of these algorithms are usually expressed in terms of computational complexity and probabilities, abstract formal models for cryptographic protocols require abstract representations of cryptographic algorithms.

In the APA protocol model presented in this paper, we model requirements on underlying cryptographic algorithms in terms of abstract secure channels. The idea of modelling communication with abstract channels was first introduced in [2, 7] and used to determine which combination of secure channels is required to realise a specific security goal in a cryptographic protocol and to compare various approaches to establish secure channels in open networks. In [1] they are used in a framework for design of secure protocols.

The use of abstract channels allows to distinguish between different properties of cryptographic algorithms while at the same time abstracting from implicit assumptions on keys and on possible implementation details. This in turn reduces the complexity of protocol specification and relocates implementation issues to a lower level of abstraction.

Our model includes abstract channels with various different properties, such as providing authenticity and proof of authenticity, indistinguishability of data, time related properties etc. However, in this paper we restrict ourselves to introducing abstract channels for authenticity and proof of authenticity and a broadcast channel (to model unprotected communication). For the formal definitions of other abstract secure channels see [10].

In the following, the S denotes the prefix closed language representing a behaviour of a system of the structure as described above. We first define send and receive events on channels. The data structure of state component Network defines that each message is tagged with the name of the channel it is sent on. So every element of the state of Network has the form *(message,channelname)*.

Definition 5 $(s_i, e_i, \bar{s}_i)$ *is a* send event *with message* m_i *on channel* ChannelName *iff* $\exists P \in \mathbb{P} : e_i = Send_P$ *and* $(m_i, ChannelName) \hookrightarrow Network(s_i)$.

Definition 6 $(s_i, e_i, \bar{s}_i)$ *is a* receive event *with message* m_i *on channel* ChannelName *iff* $\exists P \in \mathbb{P} : e_i = Rec_P$ *and* $(m_i, ChannelName) \hookleftarrow Network(s_i)$.

Authentication channel An authentication channel has the property that only one agent can send and all agents can receive messages on this channel. If agent P receives a message on Q's authentication channel, a matching send event must have happened before in all sequences that P considers to be possible.

Definition 7 *For a system* S*, a channel* $(channel, Q)$ *is called an* Authentication Channel of agent Q *if for all* $P \in \mathbb{P}$ *holds that for all* $\omega \in S$ *with the property that* $(s_i, \mathrm{Rec_P}, \bar{s}_i) \in alph(\omega)$ *is a receive event on* $(channel, Q)$ *with message* m_i*, for all* $x \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$ *exists a send event* $(s_j, e_j, \bar{s}_j) \in alph(x)$ *on* $(channel, Q)$ *with* $s_j < s_i$*,* $e_j = Send_Q$ *and message* $m_j = m_i$*. We denote the authentication channel of agent* Q *with* (Auth, Q).

Proof channel A proof channel is an authentication channel which additionally provides a proof of authenticity on receipt of a message on the channel. The proof is considered to be a message as well and can therefore be forwarded to other agents. A proof channel provides strong, transferable proofs as defined by Kailar [6].

Definition 8 *For a system* S*, a channel* $(channel, Q)$ *is called a* Proof Channel of agent Q *if*

1. $(channel, Q)$ *is an authentication channel of* Q*,*
2. *for all* $P \in \mathbb{P}$*,* $\mathcal{N} \subseteq \mathcal{M}essages$ *holds:*
   $(proof, Q, \mathcal{N}) \in \mathrm{State_P}(\bar{s}_i)$ *implies that for all* $\omega \in S$ *with* $(s_i, e_i, \bar{s}_i) \in alph(\omega)$ *send events on* $(channel, Q)$ *with message* $m \in \mathcal{N}$ *are authentic for* P *after* ω *and there exists a receive event* $(s_k, \mathrm{Rec_P}, \overline{s_k}) \in alph(\omega)$ *with* $s_k < s_i$ *and* $(proof, Q, \mathcal{N}) \hookrightarrow \mathrm{State_P}(s_k)$.
3. *for all* $P \in \mathbb{P}$ *and* $m \in \mathcal{M}essages$ *holds:*
   $(proof, Q, m) \in \mathrm{State_P}(s_i)$ *for all send events* $(s_i, \mathrm{Send_P}, \bar{s}_i)$ *with message* $m = (proof, Q, m)$.

We denote the proof channel of agent Q with $(Proof, Q)$.

Furthermore, we use a channel *broadcast* with no security properties.

5 A concrete price request protocol

In this section we present a concrete protocol implementing the price request/offer example introduced above and show that it satisfies adequate concrete representations of Property 1. To implement these properties, we use a proof channel of SP. The messages that do not require any security property are sent on a broadcast channel.

5.1 Specification

Initial state For agents $P, SP \in \mathbb{P}$ the initial state of the system is defined by $\text{Channels}_{\text{P}}(s_0) = \{(broadcast, send), ((Proof, SP), rec)\}$ and $\text{Channels}_{\text{SP}}(s_0) = \{(broadcast, rec), ((Proof, SP),\ send)\}$. If $P = SP$, then we have only one *Channels* component including the four channels given above. The state component Network and all other state components State_{P} and $\text{ApplMem}_{\text{P}}$ of agents $P \in \mathbb{P}$ are empty.

Protocol steps We use the state transition patterns introduced in Section 4.2 to specify the protocol steps.

Step 1 ()

$\xrightarrow{\textbf{Appl}_{\textbf{P}}}$

$(send, price\_req) \quad \hookrightarrow \quad \text{ApplMem}_{\text{P}}$

Step 2 ()

$(send, price\_req) \in \text{ApplMem}_{\text{P}}$
$(broadcast, send) \in \text{Channels}_{\text{P}}$
$\xrightarrow{\textbf{Send}_{\textbf{P}}}$

$(send, price\_req) \quad \hookleftarrow \quad \text{ApplMem}_{\text{P}}$
$(sent, price\_req) \quad \hookrightarrow \quad \text{State}_{\text{P}}$
$((price\_req), broadcast) \quad \hookrightarrow \quad \text{Network}$

We assume that the user does not request the price from a particular service provider, so that the message does not include any address. The user then waits for any answer some service provider may send.

Step 3 (P, m)

$(broadcast, rec) \in \text{Channels}_{\text{SP}}$
$(m, broadcast) \in \text{Network}$
$m = (price\_req)$
$\xrightarrow{\textbf{Rec}_{\textbf{SP}}}$

$(rec, P, price\_req) \quad \hookrightarrow \quad \text{ApplMem}_{\text{SP}}$

Step 4 (P)

$$
\begin{array}{l}
(rec, P, price\_req) \in \mathrm{ApplMem}_{\mathrm{SP}} \\
\xrightarrow{\mathbf{Appl_{SP}}} \\
\begin{array}{lcl}
(rec, P, price\_req) & \hookleftarrow & \mathrm{ApplMem}_{\mathrm{SP}} \\
(price, \mathit{offer}) & \hookrightarrow & \mathrm{ApplMem}_{\mathrm{SP}}
\end{array}
\end{array}
$$

Step 5 $(price)$

$$
\begin{array}{l}
(price, \mathit{offer}) \in \mathrm{ApplMem}_{\mathrm{SP}} \\
((Proof, SP), send) \in \mathrm{Channels}_{\mathrm{SP}} \\
\xrightarrow{\mathbf{Send_{SP}}} \\
\begin{array}{lcl}
(price, \mathit{offer}) & \hookleftarrow & \mathrm{ApplMem}_{\mathrm{SP}} \\
((price, \mathit{offer}), (Proof, SP)) & \hookrightarrow & \mathrm{Network}
\end{array}
\end{array}
$$

Step 6 (SP, m)

$$
\begin{array}{l}
(sent, price\_req) \in \mathrm{State}_{\mathrm{P}} \\
((Proof, SP), rec) \in \mathrm{Channels}_{\mathrm{P}} \\
(m, (Proof, SP)) \in \mathrm{Network} \\
elem(2, m) = \mathit{offer} \\
\xrightarrow{\mathbf{Rec_P}} \\
\begin{array}{lcl}
(sent, price\_req) & \hookleftarrow & \mathrm{State}_{\mathrm{P}} \\
(rec, SP, m) & \hookrightarrow & \mathrm{ApplMem}_{\mathrm{P}} \\
(proof, SP, m) & \hookrightarrow & \mathrm{State}_{\mathrm{P}}
\end{array}
\end{array}
$$

Step 7 (SP, m)

$$
\begin{array}{l}
(rec, SP, m) \in \mathrm{ApplMem}_{\mathrm{P}} \\
\xrightarrow{\mathbf{Appl_P}} \\
(rec, SP, m) \hookleftarrow \mathrm{ApplMem}_{\mathrm{P}}
\end{array}
$$

Proof Send (m, SP)

$$(proof, SP, m) \in \text{State}_\text{P}$$
$$\xrightarrow{\textbf{Send}_\textbf{P}}$$
$$((proof, SP, m), Broadcast) \quad \hookrightarrow \quad \text{Network}$$

Proof Rec (m, SP)

$$(m, broadcast) \in \text{Network}$$
$$elem(1, m) = proof$$
$$SP := elem(2, m)$$
$$\xrightarrow{\textbf{Rec}_\textbf{P}}$$
$$(proof, SP, elem(3, m)) \quad \hookrightarrow \quad \text{State}_\text{P}$$

The transition patterns Step 1, Step2 ,..., Step 7, Proof Send and Proof Rec define a set of state transitions $\mathbb{S}_{PRO}$ and a system $S_{PRO} \subseteq \Sigma^*$ without any malicious behaviour. Here Σ^* is the set of all defined state transitions in the APA protocol model described in Section 4.3. We now consider sets $S \subseteq \Sigma^*$ including not explicitly specified malicious behaviour. It is assumed that $S_{PRO} \subseteq S$. The structure of the APA model implies implicit assumptions on what agents cannot do. Most important is the following assumption. The neighborhood relationship between elementary automata and state components implies that no agent can read or change the state of other agents internal state components $\text{Channels}_\text{P}, \text{State}_\text{P}$ and ApplMem_P.

5.2 Relation between different levels of abstraction

As the next step we want to establish a relation between the abstract system S'_{PRO} specified in Section 3 and all systems S with $S_{PRO} \subseteq S \subseteq \Sigma^*$. Therefore we specify a homomorphism from Σ^* to Σ'^* and show that it preserves authenticity and proofs in accordance with Definition 3 and Definition 4, respectively. S includes malicious behaviour and is restricted by the knowledge of agents about the security mechanisms used in the system. For each agent P, this knowledge is described by the set W_P. Therefore we assume $S \subseteq W_P$ for all $P \in \mathbb{P}$.

The first assumption about the knowledge of $P \in \mathbb{P}$ concernes the proof channel of $SP \in \mathbb{SP}$.

Assumption 1 *For each $SP \in \mathbb{SP}$ the channel (Proof, SP) is a proof channel in accordance with Definition 8, i.e. for $P \in \mathbb{P}$ each $u \in W_P$ satisfies the properties of a proof channel.*

The property that every agent $P \in \mathbb{P}$ can give proof of authenticity requires that all agents keep the proofs in their respective State component. If an agent deletes proofs he cannot give proof anymore. Notice that no other agent but P itself can change State_P.

Assumption 2 *For all $P \in \mathbb{P}, SP \in \mathbb{SP}$ and messages m there is no state transition in S with $(proof, SP, m) \hookleftarrow \text{State}_\text{P}$.*

We now define a homomorphism h that maps $S \subseteq S_{PRO}$ onto the abstract system S'_{PRO}. We then show that this homomorphism preserves proof pairs.

Definition 9 *Let $S \subseteq \Sigma^*$ be the protocol system defined by the state transition patterns in Section 5 enriched by malicious behaviour and let $S'_{PRO} \subseteq \Sigma'^*$ be the abstract system defined in Section 3. We define a homomorphism $h : \Sigma^* \longrightarrow \Sigma'^*$ for $P \in \mathbb{P}$ and $SP \in \mathbb{SP}$ by*

$$h(s,(e,a,var),\bar{s}) = \begin{cases} \text{OFFER-S}_{\text{SP}} & if \quad e = \text{Send}_{\text{SP}} \text{ and} \\ & \quad (m,(Proof,SP)) \hookrightarrow Network \\ & \quad with\ elem(2,m) = \text{offer} \\ \text{PROOF-R}_{\text{P}}(SP) & if \quad e = \text{Rec}_{\text{P}} \text{ and } (proof,SP,m) \hookrightarrow \text{State}_{\text{P}} \\ & \quad with\ elem(2,m) = \text{offer} \\ \text{PROOF-S}_{\text{P}}(SP) & if \quad e = \text{Send}_{\text{P}} \text{ and} \\ & \quad ((proof,SP,m),broadcast) \hookrightarrow \text{Network} \\ & \quad with\ elem(2,m) = \text{offer} \\ \varepsilon & else \end{cases}$$

$(s,(e,a,var),\bar{s})$ denotes a state transition of elementary automaton e with $a \in \phi_e$ and the list of interpretations of variables var.

Homomorphism h is now used to show that the concrete protocol including malicious behaviour satisfies an adequate representation of Property 1 of the abstract specification S'_{PRO} in Section 3. First, it is shown that the homomorphic image of S under homomorphism h is the abstract specification S'_{PRO} assuming the use of secure proof channels. Then, h is proven to preserve authenticity and proofs. This section concludes with a proof that the system S_{PRO} enriched with malicious behaviour satisfies a property analogous to Property 1, as it provides the respective proof pairs.

Lemma 1 *If Assumption 1 holds then $h(W_P) \subseteq W'_P$ for $P \in \mathbb{P}$.*

Proof: We show that for every sequence $u \in (\Sigma')^*$ that is not in W'_P all sequences in the inverse images $h^{-1}(u)$ violate the properties of the proof channel and consequently are not in W_P. $u \notin W'_P$ implies

$$u \in \Big(\bigcup_{SP \in \mathbb{SP}, Q \in \mathbb{P}} \quad (\Sigma' \setminus \{\text{OFFER-S}_{\text{SP}}\})^*\{\text{PROOF-R}_{\text{Q}}(SP)\}\Sigma'^* \\ \cup(\Sigma' \setminus \{\text{PROOF-R}_{\text{Q}}\})^*\{\text{PROOF-S}_{\text{Q}}(SP)\}\Sigma'^* \ \Big)$$

By definition of h and Definition 8 follows that each state transition sequence in $h^{-1}(u)$ violates the properties of (Proof,SP), thus is not element of W_P. This is a contradiction to Assumption 1 and it follows that $h(W_P) \subseteq W'_P$. □

By induction over the sequences of state transitions in S_{PRO} it can be shown that

Lemma 2 *For $S_{PRO} \subseteq \Sigma^*$ defined by the transition patterns Step 1, Step2 ,..., Step 7, Proof Send and Proof Rec holds: $h(S_{PRO}) = S'_{PRO}$.*

Lemma 3 *$h(S) = S'_{PRO}$ if $S_{PRO} \subseteq S \subseteq W_P$ for all $P \in \mathbb{P}$.*

Proof: By $S \subseteq W_P$ and Lemma 1 holds $h(S) \subseteq h(W_P) \subseteq W'_P = S'_{PRO}$. With $h(S_{PRO}) = S'_{PRO}$ and $S \supseteq S_{PRO}$ follows $h(S) = S'_{PRO}$. □

In order to show that h preserves authenticity and proofs we need to specify the local view of the agents on both levels of abstraction. As already explained in Section 2.2, the agents' local view of the abstract system is given by $\pi'_P : \Sigma'^* \to \Sigma'^*{}_{/P}$ with $\pi'_P(x) = x$ if $x \in \Sigma'_{/P}$ and $\pi'_P(x) = \varepsilon$ if $x \in \Sigma' \setminus \Sigma'_{/P}$, i.e $\lambda'_P = \pi'_P$.

The following gives a general definition of the agents' local view in a system modelled by APA.

Definition 10 *Let $P \in \mathbb{P}$ be an agent, E_P the set of elementary automata of P and s be a global system state of a state transition system $S \subseteq \Sigma^*$. Then $s_P = \bigtimes_{C \in N(E_P)}(C(s))$ defines P's view of the state s. The agent's view of the system S is given by the homomorphism $\lambda_P : \Sigma^* \longrightarrow \Sigma_P^*$ with*

$$\lambda_P((s, e, \bar{s})) = \begin{cases} (s_P, e, \bar{s}_P) & \text{if } e \in E_P \\ \varepsilon & \text{else} \end{cases}$$

Lemma 4 *h preserves authenticity on S.*

Proof: According to Definition 3, for each $P \in \mathbb{P}$ we need to find a homomorphism $h'_P : \lambda_P(S) \longrightarrow \lambda'_P(S'_{PRO})$ such that $\lambda'_P \circ h = h'_P \circ \lambda_P$. This homomorphism is defined by

$$h'_P(s_P, (e, a, var), \bar{s}_P) = \begin{cases} \text{PROOF-R}_\text{P}(SP) & \text{if } e = \text{Rec}_\text{P} \\ & \text{and } (proof, SP, m) \hookrightarrow \text{State}_\text{P} \\ & \text{with } elem(2, m) = offer \\ \text{PROOF-S}_\text{P}(SP) & \text{if } e = \text{Send}_\text{P} \text{ and} \\ & ((proof, SP, m), broadcast) \hookrightarrow \text{Network} \\ & \text{with } elem(2, m) = offer \\ \varepsilon & \text{else} \end{cases}$$

Obviously, this homomorphism provides the desired property. □

Lemma 5
For $SP \in \mathbb{SP}$ let $\Gamma\text{S}'_{SP} = \{\text{PROOF-S}_\text{P}(SP) | P \in \mathbb{P}\}$ and $\Gamma\text{P}'_{SP} = \{\text{PROOF-R}_\text{P}(SP) | P \in \mathbb{P}\}$. Then for each $SP \in \mathbb{SP}$ the homomorphism $h : \Sigma^* \longrightarrow \Sigma'^*$ defined in Definition 9 preserves $(\Gamma\text{S}'_{SP}, \Gamma\text{P}'_{SP})$ Proofs on S.

Proof: By Lemma 4 homomorphism h preserves authenticity on S. Thus it remains to show conditions 1 and 2 of Definition 4.

Condition 1 We show the implication in two steps.

(i)
We first show $h(u)^{-1}(h(S)) \cap (\Gamma\text{S}'_{SP} \cap \Sigma'_{/P}) \neq \emptyset$ implies $u^{-1}(S) \cap (h^{-1}(\Gamma\text{S}'_{SP}) \cap \Sigma_{/P}) \neq \emptyset$: In S_{PRO}, an action PROOF-S$_\text{P}(SP)$ cannot occur without a previous PROOF-R$_\text{P}(SP)$. By Lemma 3 holds $h(S) = S'_{PRO}$. Therefore, $h(u)^{-1}(h(S)) \cap (\Gamma\text{S}'_{SP} \cap \Sigma'_{/P}) \neq \emptyset$ implies $alph(h(u)) \cap (\Gamma\text{P}'_{SP} \cap \Sigma'_{/P}) \neq \emptyset$. By the definition of h follows $alph(u) \cap (h^{-1}(\Gamma\text{P}'_{SP} \cap \Sigma'_{/P})) \neq \emptyset$. Hence, in u there exists a receive action by P with $(proof, SP, m) \hookrightarrow \text{State}_\text{P}$. Now, with Assumption 2 follows that this proof is not removed, i.e. $(proof, SP, m) \in \text{State}_\text{P}$ after u has happened. Therefore, a state transition $a \in \Sigma$ as defined with transition pattern Proof Send is possible, hence $u^{-1}(S) \cap (h^{-1}(\Gamma\text{S}'_{SP}) \cap \Sigma_{/P}) \neq \emptyset$.

(ii)
$a \in \Gamma\text{S}'_{SP}) \cap \Sigma_{/P}$ implies that $((proof, SP, m), broadcast) \in$ Network after ua has happened. Therefore, a state transition $b \in \Sigma$ in accordance with transition pattern Proof Rec is possible, hence $u^{-1}(S) \cap (h^{-1}(\Gamma\text{P}'_{SP}) \cap \Sigma_{/R}) \neq \emptyset$.
This shows that u can be continued in S with appropriate Proof Send and Proof Receive actions.

Condition 2 Holds by definition of h. □

Property 2 *For* $SP \in \mathbb{SP}$ *let* $\Gamma S_{SP} = h^{-1}(\Gamma S'_{SP}) \cap \Sigma$, $\Gamma P_{SP} = h^{-1}(\Gamma P'_{SP}) \cap \Sigma$ *and* $\Gamma_{SP} = (h^{-1}(\text{OFFER-S}_{\text{SP}}) \cap \Sigma$. *Then for each* $SP \in \mathbb{SP}$, $(\Gamma S_{SP}, \Gamma P_{SP})$ *is a proof action pair of authenticity for* Γ_{SP} *with respect to* $(W_P)_{P \in \mathbb{P}}$.

Proof: By Theorem 2, the assertion holds if the following conditions hold:

(1) $(\Gamma S'_{SP}, \Gamma P'_{SP})$ is a proof action pair of authenticity for $\{\text{OFFER-S}_{\text{SP}}\}$.
This holds by Property 1 in Section 3.

(2) Homomorphism h preserves $(\Gamma S'_{SP}, \Gamma P'_{SP})$-Proofs on S.
This condition holds by Lemma 5. □

6 Conclusion

In this paper we have presented a methodology for the specification of e-commerce transactions on different levels of abstraction, providing certain security properties. Based on a formal framework for binding cooperations we have defined the concepts of authenticity and proof of authenticity, using the notions of formal languages and language homomorphisms. The universality of the definitions allows to apply them to any specification language with a semantics based on label transition systems. We have formulated conditions on homomorphisms under which they preserve these properties from a higher to a lower abstraction level, thus serving as a means of refinement. This is in line with our general approach for verification of communicating systems where arbitrary safety and liveness properties of abstract specifications are transfered to more concrete ones by means of so-called simple language homomorphisms [9, 8]. For the specification on a lower abstraction level, we have used Asynchronous Product Automata (APA), a general class of communicating automata. Suitable homomorphisms map an APA specification onto a more abstract specification and transfer authenticity and proofs from the higher to the lower abstraction level.

Our approach is related to a range of work concerning authentication and proof (see, for example, [12], [13] and [6]). However, these papers are concerned with the analysis of security protocols. To the best of our knowledge, there is no approach that formally defines a proof of authenticity and provides a design methodology. Moreover, our approach is equally useful for the continuation of the design process where the protocol specification is transfered to the next level of refinement by introducing cryptographic mechanisms. This is subject of future work.

In a forthcoming paper it will be shown that our approach is adequate to formalize the concept of confidentiality.

References

[1] C. Boyd. A Framework for Design of Key Establishment Protocols. *Lecture Notes in Computer Science*, 1172:146–157, 1996.

[2] C. Boyd and W. Mao. Design and analysis of key exchange protocols via secure channel identification. In J.Pieprzyk and R. Safavi-Naini, editors, *ASIACRYPT '94*, volume 917 of *Lecture Notes in Computer Science*, pages 171–181. Springer, 1994.

[3] E.M. Clarke, S. Jha, and W. Marrero. A machine checkable logic of knowledge for specifying security properties of electronic commerce protocols. In *LICS Security Workshop*, 1998.

[4] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[5] R. Grimm and P. Ochsenschläger. Binding cooperation, a formal model for electronic commerce. *Computer Networks*, 37:171–193, 2001.

[6] R. Kailar. Accountability in Electronic Commerce Protocols. In *IEEE Transactions on Software Engineering*, volume 22, pages 313–328. IEEE, 1996.

[7] U. M. Maurer and P. E. Schmid. A calculus for secure channel establishment in open networks. In Dieter Gollmann, editor, *Computer Security – ESORICS 94*, volume 875 of *LNCS*, pages 175 – 192. Springer Verlag, November 1994.

[8] P. Ochsenschläger, J. Repp, and R. Rieke. Abstraction and a verification method for co-operating systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 12:447–459, June 2000.

[9] P. Ochsenschläger, J. Repp, R. Rieke, and U. Nitsche. The SH-Verification Tool Abstraction-Based Verification of Co-operating Systems. *Formal Aspects of Computing, The International Journal of Formal Method*, 11:1–24, 1999.

[10] C. Rudolph. *A Model for Secure Protocols and its Application to Systematic Design of Cryptographic Protocols*. PhD thesis, Queensland University of Technology, 2001.

[11] S. Schneider. Modelling Security Properties with SCP. Technical Report CSD-TR-96-14, Royal Holloway, 1996.

[12] S. Schneider. Security Properties and CSP. In *Symposion on Security and Privacy*. IEEE, 1996.

[13] P. Syverson and C. Meadows. A Logical Language for Specifying Cryptographic Protocol Requirements. In *Proceedings of the 1993 IEEE Computer Society Symposium on Security and Privacy*, pages 165–177. IEEE Computer Society Press, New York, 1993.

A Appendix

An additional class of actions $\Gamma\mathrm{L} \subseteq \Sigma$ describes loss of proofs: If agent P has executed an action from $\Gamma\mathrm{L}$ then he has lost his proofs of authenticity for $\Gamma \subseteq \Sigma$. This implies the following modifications of Definition 2, Definition 4, and Theorem 2:

Definition A.2 (Proof of authenticity) *A tripel* $(\Gamma\mathrm{S}, \Gamma\mathrm{P}, \Gamma\mathrm{L})$ *with* $\Gamma\mathrm{S} \subseteq \Sigma$, $\Gamma\mathrm{P} \subseteq \Sigma$, *and* $\Gamma\mathrm{L} \subseteq \Sigma$ *is a triple of sets of proof actions of authenticity for a set* $\Gamma \subseteq \Sigma$ *on* S *with respect to* $(W_P)_{P\in\mathbb{P}}$ *if for all* $\omega \in S$ *and for all* $P \in \mathbb{P}$ *the following holds:*

1. *For* P *the set* Γ *is authentic after* ω *if* $alph(\pi_P(\omega)) \cap \Gamma\mathrm{P} \neq \emptyset$.
2. *For each* $R \in \mathbb{P}$ *there exists actions* $a \in \Sigma_{/P} \cap \Gamma\mathrm{S}$ *and* $b \in \Sigma_{/R} \cap \Gamma\mathrm{P}$ *with* $\omega ab \in S$ *if and only if* $\omega \in \Sigma^*(\Gamma\mathrm{P} \cap \Sigma_{/P})(\Sigma \setminus (\Gamma\mathrm{L} \cap \Sigma_{/P}))^*$.

Agent $P \in \mathbb{P}$ *can give proof of authenticity of* $\Gamma \subseteq \Sigma$ *after a sequence of actions* $\omega \in S$ *if for each* $R \in \mathbb{P}$ *there exists actions* $a \in \Sigma_{/P} \cap \Gamma\mathrm{S}$ *and* $b \in \Sigma_{/R} \cap \Gamma\mathrm{P}$ *with* $\omega ab \in S$.

In the following, we shortly call $(\Gamma S, \Gamma P, \Gamma L)$ a *proof action triple for* Γ.

Definition A.4 *Let* $\Gamma S' \subseteq \Sigma'$, $\Gamma P' \subseteq \Sigma'$, *and* $\Gamma L' \subseteq \Sigma'$. $h : \Sigma^* \to \Sigma'^*$ *preserves* $(\Gamma S', \Gamma P', \Gamma L')$*-proofs on* S *if* h *preserves authenticity on* S *and if the following holds:*

1. $h(u)^{-1}(h(S)) \cap ((\Gamma S' \cap \Sigma'_{/P})(\Gamma P' \cap \Sigma'_{/R})) \neq \emptyset$ *implies* $u^{-1}(S) \cap ((h^{-1}(\Gamma S' \setminus \Sigma'_{/\#}) \cap \Sigma_{/P})(h^{-1}(\Gamma P' \setminus \Sigma'_{/\#}) \cap \Sigma_{/R})) \neq \emptyset$ *for each* $P, R \in \mathbb{P}$ *and* $u \in S$.
2. $h(\Sigma_{/P}) \cap (\Sigma' \setminus \Sigma'_{/\#}) \subseteq \Sigma'_{/P}$ *for each* $P \in \mathbb{P}$ *and* $h(\Sigma_{/\#}) \subseteq \Sigma'_{/\#}$.

Theorem A.2 *Let* $(\Gamma S', \Gamma P', \Gamma L')$ *be a proof action triple for* $\Gamma' \subseteq \Sigma'$ *on* $h(S)$ *with respect to* $(W'_P)_{P \in \mathbb{P}}$. *Let* $h : \Sigma^* \longrightarrow (\Sigma')^*$ *be a homomorphism that preserves* $(\Gamma S', \Gamma P', \Gamma L')$*-proofs on* S*, and let* $\Gamma S = h^{-1}(\Gamma S' \setminus \Sigma'_{/\#}) \cap \Sigma$, $\Gamma P = h^{-1}(\Gamma P' \setminus \Sigma'_{/\#}) \cap \Sigma$, $\Gamma L = h^{-1}(\Gamma L' \setminus \Sigma'_{/\#}) \cap \Sigma$, *and* $\Gamma = h^{-1}(\Gamma') \cap \Sigma$. *Then:*

1. $(\Gamma S, \Gamma P, \Gamma L)$ *is a proof action triple for* $\Gamma \subseteq \Sigma$ *on* S ,
2. *if agent* $P \in \mathbb{P}$ *can give proof of authenticity of* Γ' *after* $\omega' \in h(S)$ *then* P *can give proof of authenticity of* Γ *after each* $\omega \in h^{-1}(\omega') \cap S$

with respect to each family $(W_P)_{P \in \mathbb{P}}$ *satisfying* $W_P \subseteq h^{-1}(W'_P)$ *for all* $P \in \mathbb{P}$.

Proof:

To prove proposition 1, we first show authenticity of Γ for P after ω. If $\omega \in S$ and $alph(\pi_P(\omega)) \cap \Gamma P \neq \emptyset$ then $h(\omega) \in h(S)$ and $alph(\pi'_P(h(\omega))) \cap \Gamma P' \neq \emptyset$ by the second condition of Definition A.4. Now authenticity of Γ for P after ω follows from authenticity of Γ' for P after $h(\omega)$ by Theorem 1.

Now we prove the second condition of Definition A.2. We first show that in S a sequence ω contains no loss action after a proof receive action if and only if this holds in the abstract system $h(s)$. This follows from the second condition of Definition A.4 that states in principle that actions of agent P that are not mapped to $\Sigma_{/\#}$ have to be mapped onto actions of agent P in the abstract level. Thus it follows $\omega \in S \cap \Sigma^*(\Gamma P \cap \Sigma_{/P})(\Sigma \setminus (\Gamma L \cap \Sigma_{/P}))^*$ if and only if $h(\omega) \in h(S) \cap \Sigma'^*(\Gamma P' \cap \Sigma'_{/P})(\Sigma' \setminus (\Gamma L' \cap \Sigma'_{/P}))^*$.

We now show that ω can be continued with send and receive actions in S if and only if the respective statement holds for the image in $h(S)$. The first part of the equivalence, namely $h(\omega)^{-1}(h(S)) \cap ((\Gamma S' \cap \Sigma'_{/P})(\Gamma P' \cap \Sigma'_{/R})) \neq \emptyset$ implies $\omega^{-1}(S) \cap ((h^{-1}(\Gamma S' \setminus \Sigma'_{/\#}) \cap \Sigma_{/P})(h^{-1}(\Gamma P' \setminus \Sigma'_{/\#}) \cap \Sigma_{/R})) \neq \emptyset$, follows from the first condition of Definition A.4. The other direction of the equivalence, $\omega^{-1}(S) \cap ((h^{-1}(\Gamma S' \setminus \Sigma'_{/\#}) \cap \Sigma_{/P})(h^{-1}(\Gamma P' \setminus \Sigma'_{/\#}) \cap \Sigma_{/R})) \neq \emptyset$ implies $h(\omega)^{-1}(h(S)) \cap ((\Gamma S' \cap \Sigma'_{/P})(\Gamma P' \cap \Sigma'_{/R})) \neq \emptyset$, follows again from the second condition of Definition A.4.

Now with the equivalence of $\omega^{-1}(S) \cap ((h^{-1}(\Gamma S' \setminus \Sigma'_{/\#}) \cap \Sigma_{/P})(h^{-1}(\Gamma P' \setminus \Sigma'_{/\#}) \cap \Sigma_{/R})) \neq \emptyset$ and $h(\omega)^{-1}(h(S)) \cap ((\Gamma S' \cap \Sigma'_{/P})(\Gamma P' \cap \Sigma'_{/R})) \neq \emptyset$, the assumption that $(\Gamma S, \Gamma P, \Gamma L)$ is a proof action triple implies the second condition of Definition A.2 , which completes the proof of 1. Proposition 2 is shown similarily. □