On a formal framework for security properties

Sigrid Gürgens, Peter Ochsenschläger, Carsten Rudolph

Fraunhofer Institute for Secure Information Technology SIT
Rheinstrasse 75, D-64295 Darmstadt, Germany

Abstract

A new approach to property-based characterisation of security requirements is presented. The main goal is to provide a framework for the specification of a wide variety of security requirements with formal semantics in terms of security properties of a discrete model of a system. In contrast to previous approaches it is not focused on a special type of security property. The classical concept of "properties" comprising safety and liveness properties is extended to include security properties. Formalisations of authenticity, different types of non-repudiation and confidentiality are presented within the framework. Several examples illustrate the flexibility of this approach.

Key words: Security requirements, formal security models, security properties

1 Introduction

Nowadays it is no longer necessary to accentuate the need for security in all types of computing systems. It is widely accepted that security issues are essential in particular for tele-cooperation systems, mobile communication systems or other types of distributed systems, and that security needs to be considered in all phases of the development cycle of such systems. This paper concentrates

Email address: `rudolphc@sit.fraunhofer.de` (Carsten Rudolph).

on the requirements specification phase of the development process. A wide variety of approaches to security requirements specification has been developed. Among these one can distinguish two main approaches. In the first, a security model is used to express either an abstract view of a particular desired (secure) behaviour of a system or to specify undesired behaviour. In the second approach, formally characterised security properties are used to specify security requirements. One classical approach of the first category is the multi-level security model by Bell and La Padula [2]. The Bell-LaPadula model and other related security models, like the more recent work by McLean [11], define models for access control and therefore do not meet all kind of security requirements of complex distributed or tele-cooperating systems. The integrity model by Clark and Wilson [5] formulates restrictions on how data items might be altered. Other approaches are based on the specification of misuse cases describing threats by malicious agents or threat scenarios [9].

Concerning security properties, a large part of research work concentrates on information flow control and non-interference. The underlying trace based system models for some of these notions [15, 10] are closely related to the model presented in this paper. Non-interference and information flow properties can be used to describe various confidentiality properties. The underlying formal models are often highly complex and therefore accurate confidentiality requirements specification is error-prone and difficult. Another emphasis of research on security properties lies on authenticity and non-repudiation. Starting with the work on authentication logics [3], a wide variety of notions of authenticity has been published. Most of the formalisations are tailored for special cases like authentication protocol analysis while others provide more general definitions [13]. A wide variety of other fomalisations of security properties has been proposed, but there exists no framework in which different security requirements can be specified based on a single representation of a system.

In this paper we present a new approach to property-based characterisation of security requirements. In contrast to previous approaches it is not focused on a special type of security property. The main goal is to provide a framework for the specification of a wide variety of security requirements with formal semantics in terms of security properties of a single discrete model of the system. In addition to security requirements specification, property preserving abstractions by alphabetic language homomorphisms are used to transport the security properties from higher to lower levels of abstraction. We show that previously published formal definitions of authenticity and proof of authenticity [6b] and parameter confidentiality [6c] provide the foundation for the specification of many useful security properties.

The underlying formal model describes system behaviours as (sets of) traces of actions, where these actions are associated with agents in the systems. This type of specification is very common, but for security properties additional information is required. First, satisfaction of security properties depends on the agents' view of the system. In our framework, this view has to be specified for each system as an alphabetic language homomorphism on traces of actions. Agents' views are used in other approaches as well (e.g. by Wedel and Kessler for the semantics of the authentication logic AUTLOG [14], or by Heisel et al. [7], defining the window to a system. However, agents' views in our approach are more flexible, as will be demonstrated in Section 3. Second, for each agent the knowledge about the global system has to be part of the system specification. Obviously, satisfaction of confidentiality depends on the initial knowledge of the attacker. Furthermore, trust on underlying security mechanisms, such as cryptographic algorithms, are describe as knowledge about the system. Although satisfaction of security properties depends on the particular knowledge of agents about the system, this is neglected in all existing system models for security properties.

Our framework was used for the design of secure e-commerce protocols in the project CASENET [4] funded by the European Commission.

In this paper, we first describe and explain our model framework based on formal languages. This framework is transparent with respect to any particular notion of system actions. Formal definitions for authenticity, proof of authenticity and parameter confidentiality have been previously proposed for such a framework [6b, 6c]. We present a variety of possible instantiations in order to demonstrate the flexibility of the approach. Further, for every definition we determine the parameters necessary to accurately specify a particular security requirement. Examples illustrate that the formalisations are both accurate and understandable.

2 System behaviour specification and agents' knowledge about a system

The *behaviour* S of a discrete system can be formally described by the set of its possible sequences of actions (traces). Therefore $S \subseteq \Sigma^*$ holds where Σ is the set of all actions of the system, and Σ^* is 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, 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 $\text{pre}(u) \subseteq S$ 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* $u^{-1}(S) = \{y \in \Sigma^* \mid uy \in S\}$.

Classical liveness and safety properties can easily be specified for such a system using well known formalisations. For security properties, we need to extend the system model by taking into account the agents' view of the system and agents' knowledge about the global system behaviour.

2.1 Agents' view and knowledge about the global system behaviour

Security properties can only be satisfied relative to particular sets of underlying system assumptions. Examples include assumptions on cryptographic algorithms, secure storage, trust in the correct behaviour of agents or reliable data transfer. Relatively small changes in these assumptions can result in huge differences concerning satisfaction of security properties. Every model for secure systems must address these issues. However, most existing models rely on a fixed set of underlying assumptions (see for example [3] and [12]). Most of these assumptions are often implicitly given by particular properties of the model framework. Thus, it is very hard to verify whether a particular implementation actually satisfies all of these assumptions. Further, imprecise security assumptions might result in correct but useless security proofs and finally in insecure implementations. Therefore, a model for secure systems needs to provide the means to accurately specify underlying system assumptions in a flexible way.

In order to provide the required flexibility, we extend the system specification by two components: *agents' knowledge* about the global system behaviour and *agents' view*. The knowledge about the system consists of all traces that an agent initially considers possible, i.e. all traces that do not violate any system assumptions, and the view of an agent specifies which parts of the system behaviour the agent can actually see. In the following paragraphs, these two components and their relations are explained in detail.

For any agent P its knowledge about the global system behaviour $W_P \subseteq \Sigma^*$ is considered to be part of the system specification.

We may assume for example that a message that was received must have been sent before. Thus an agent's W_P will contain only those sequences of actions in which a message is first sent and then received. All sequences of actions included in W_P in which a digital signature is received and verified by using some agent Q's public key will contain an action where Q generated this signature.

Care must be taken when specifying the sets W_P for all agents P in order not to specify properties that are desirable but not guaranteed by verified system assumptions. In a setting for example where we assume one time passwords are used, if P trusts Q, W_P contains only those sequences of actions in which Q sends a certain password only once. However, if Q cannot be trusted, W_P will also contain sequences of actions in which Q sends a password more than once. In Section 3 we will give an example of W_P.

The specification of the desired system behaviour generally does not include behaviour of malicious agents which has to be taken into account in open systems. An approach which is frequently used for the security analysis of cryptographic protocols is to extend the system specification by explicit specification of malicious 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. In our approach, the explicit specification of agents' knowledge about system and environment allows to discard explicit specification of malicious behaviour. Every behaviour which is not explicitly excluded by some W_P is allowed. Denoting a system containing malicious behaviour by S and the correct system behaviour by S_C, we assume $S_C \subseteq S \subseteq \Sigma^*$. We further assume $S \subseteq W_P$, i.e. every agent considers the system behaviour to be possible. Security properties can now be defined relative to W_P. The relation between the system behaviour without malicious actions S_C, the system behaviour including malicious actions S, and W_P is graphically shown in Figure 1.

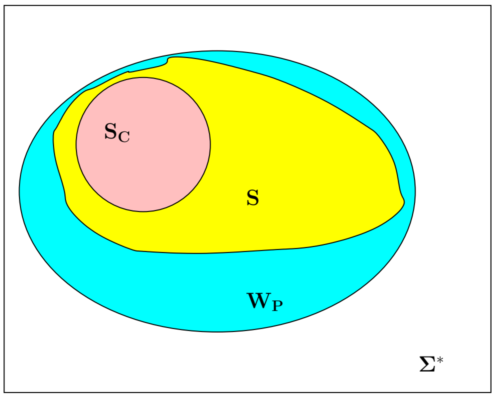

Fig. 1. System behaviour and W_P

The set W_P describes what P knows initially. However, in a running system P can learn from actions that have occurred. Satisfaction of security properties obviously also depends on what agents are able to learn. After a sequence of actions $\omega \in S$ has happened, every agent can use its *local view* of ω to determine the sequences of actions it considers to be possible. In order to determine what is the local view of an agent, we first assign every action to exactly one agent. Thus $\Sigma = \dot{\bigcup}_{P\in\mathbb{P}}\Sigma_{/P}$ (where $\Sigma_{/P}$ denotes all actions performed by agent P, and $\dot{\bigcup}$ denotes the disjoint union). 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.

The projection π_P is the correct representation of P's view of the system if all information about an action $x \in \Sigma_{/P}$ is available for agent P and P can only see its own actions. In this case P's local view of the sequence of actions $\omega = (\mathit{send}, P, m1)(\mathit{rec}, Q, m1)$ for example is $(\mathit{send}, P, m1)$. However, P's view may be finer. For example it may additionally note other agents' actions without seeing the messages sent and received, respectively. In this case, P's local view of ω will be equal to $(\mathit{send}, P, m1)(\mathit{rec}, Q)$. P's local view may also be coarser than π_P. In a system the actions of which are represented by a triple (*global state, transition label, global successor state*), although seeing its own actions, P will not be able to see the other agents' state.

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 contain all information about the system behaviour S.

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. In the above example with the projection on P being P's local view, $\lambda_P^{-1}(\lambda_P(\omega))$ consists of sequences each of which contains an action $(\mathit{send}, P, (Q, m1))$. For some agent R that does not take part in ω, $\lambda_R^{-1}(\lambda_R(\omega))$ consists of sequences of actions of other agents, i.e. is equal to $(\Sigma \setminus \Sigma_{/R})^*$.

Depending on its knowledge about the system S, underlying security mechanisms and system assumptions, P does not consider all sequences in $\lambda_P^{-1}(\lambda_P(\omega))$ possible. Thus it can use its 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. The set $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$ is similar to the possible worlds semantics that have been defined for authentication logics in the context of cryptographic protocols [1, 14]. Our notion is more general because for authentication logics λ_P and W_P are fixed for all systems, whereas in our approach they can be defined differently for different systems. The knowledge of P relative to a sequence of actions ω is graphically shown in Figure 2.

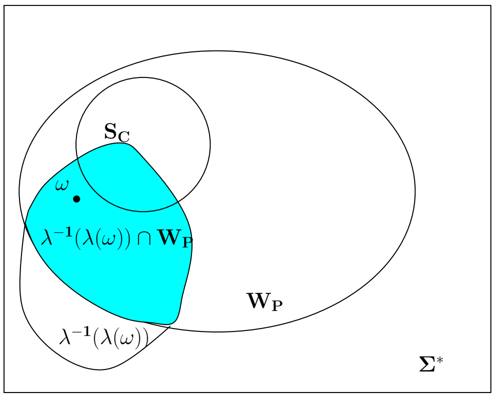
$\mathbf{S_C}$
ω
$\lambda^{-1}(\lambda(\omega)) \cap \mathbf{W_P}$
$\mathbf{W_P}$
$\lambda^{-1}(\lambda(\omega))$
$\mathbf{\Sigma}^*$

Fig. 2. Local view of P

3 Security requirements specification and examples

We first introduce a short example which is used throughout the remainder of the paper to illustrate our definitions of authenticity, proof of authenticity, and parameter confidentiality, and to specify various security requirements based on these definitions. Please note that the example is artificial and not supposed to specify appropriate security requirements for any real life application. In the project CASENET [4] funded by the European Commission we demonstrated that our approach can be successfully applied to real life e-government and e-business applications.

Action	Explanation
(send-login,U_i,(username,password))	U_i starts a new session by sending the login information.
(rec-login,SP,(U_i,username,password))	SP receives and verifies login information by U_i.
(send-request,U_i,(request-data))	After starting the session, U_i sends a request.
(rec-request,SP,(U_i,request-data))	SP receives and processes the request by U_i.
(send-result,SP,(U_i,result-data))	SP sends the result to U_i.
(rec-result,U_i,(result-data))	U_i receives the result.
(send-logoff,U_i,())	U_i terminates the session.
(rec-logoff,SP,(U_i))	SP terminates the session with U_i.

Table 1
Actions for the example system

The system for our example consists of users $U_1, \ldots, U_n$ and a service provider SP. It is assumed that a password is required to access the service.

Table 1 shows all possible actions in our example system and the agents associated with these actions. The format of actions is *(action,agent,parameters)*, where *action* identifies the action, *agent* denotes the agent the action is associated with and *parameters* includes the message and other parameters of the action.

Malicious agents can act using arbitrary unspecified actions (restricted by security assumptions, security mechanisms, ...).

The actions as they are presented in Table 1 are specified on a very high level of abstraction, suitable for high level system specification. However, security requirements for finer-grained systems specification with much more complex actions (e.g state transitions including information of the system state) can be specified similar to the abstract examples shown in the following sections.

The local view λ_P of the agents in our system for those examples concerned with authenticity is simply the projection on their respective actions. For the confidentiality examples, we will consider different local views (see Section 3.5.2). As to agents' knowledge about the system, we consider two different sets W_P^1 and W_P^2 (all agents P know the same about the system). Both sets do not contain sequences of actions with a message being received without having been sent. Additionally, W_P^1 does not contain sequences where (rec-login,*SP*, (U_i,username,password)) has happened without an action (send-login,U_i,(username,password)) before, while W_P^2 contains such sequences, all send login actions that happen before (rec-login,*SP*,(U_i,username,password)) being performed by some agent $U_j \neq U_i$.

To give the reader an idea of how W_P specification can be done, we formally specify

$$\boldsymbol{W_P^1} = \Sigma^* \setminus \bigcup_{\substack{U_i \in \mathbb{P} \\ n,p \in \boldsymbol{M}}} \Big[(\Sigma \setminus \{(\text{send-login},U_i,(\text{n},\text{p}))\})^* \{(\text{rec-login},SP,(U_i,\text{n},\text{p}))\} \Sigma^* \cup \mathcal{R} \Big]$$

where $\mathcal{R}$ contains other sequences of actions not considered possible by the agents. We do not go into more detail here.

In the following paragraphs we present and explain the formal definitions, which are then illustrated using the example introduced above. The different specification of agents' knowledge W_P of the system and agents' local view λ_P show that these are important in order to determine whether or not a certain security property holds. Thus for the specification of a real life application and its security requirements, care must be taken when specifying λ_P and W_P.

3.1 Authenticity

3.1.1 Definition of the property

In the context of sequences of actions, authenticity is the authenticity of a particular action. Since usually authenticity of some action for a certain agent is required, the definition has to refer in some way to the agent. Thus we call a particular action $a \in \Sigma$ authentic for agent P if in all sequences that P considers possible after a sequence of actions ω has happened, some time in the past a must have happened. By extending this definition to a set of actions Γ being authentic for P if one of the actions in Γ is authentic for P we gain the flexibility that P does not necessarily need to know all parameters of the authentic action. For example, a message may consist of one part protected by a digital signature and another irrelevant part without protection. Then, the recipient can know that the signer has sent a message containing the signature, but the rest of the message is not authentic. Therefore, in this case, Γ comprises all messages containing the relevant signature and arbitrary other message parts.

The formal definition for authenticity is as follows.

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 $x \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$.*

This definition and the following definition for proof of authenticity was presented in [6b]. A similar definition of authenticity has been previously proposed by Schneider [13] who defines authenticity of events or sets of events on globally viewed traces.

3.1.2 Examples

- The sending of the login message by U_i shall be authentic for SP each time it executes action (rec-login,SP,(U_i,username,password)). Thus the set Γ of actions that shall be authentic consists of only the action (send-login,U_i,(username,password)), the agent P for which this action shall be authentic is SP, and it shall be authentic for all sequences of actions ω that contain an action (rec-login,SP,(U_i,username,password)). Formally, this is denoted by requesting for all $\omega \in S$ that (rec-login,SP,(U_i,username,password)) $\in alph(\omega)$ implies $alph(x) \cap \{$(send-login, U_i, (username,password))$\}$

$\neq \emptyset$ for all $x \in \lambda_{SP}^{-1}(\lambda_{SP}(\omega)) \cap W_{SP}$. This security property is provided by a system with agents' knowledge equal to W_P^1, but does not hold if agents' knowledge is specified as W_P^2.

- The sending of an arbitrary send-request by U_i shall be authentic to *SP*. We only require that *request-data* can be any element in a subset *Requests* of the messages. This property expresses that *SP* knows that U_i has send a request, but does not know whether the request-data sent by U_i is exactly the same data that *SP* has received in its rec-request action. Thus $\Gamma = \{$(send-request,U_i,(request-data))$|$ request-data $\in$ *Requests*$\}$ shall be authentic for $P = SP$, and we require authenticity for all ω that contain (rec-request,*SP*,(U_i,n)).

  Note that this requirement can be too weak for many real life cases, because *SP* cannot conclude that n=request-data, i.e. *SP* knows that U_i has sent some message, but does not know which. Furthermore, if *SP* knows that certain request-data m is never sent by U_i, i.e. if W_{SP} does not contain sequences ω with (send-request,U_i,(m)) $\in alph(\omega)$, Γ is never authentic for *SP* after having performed the action (rec-request,*SP*,(U_i, m)).
- The receipt of the result performed by U_i shall be authentic to *SP*. Here we require that $\Gamma = \{$(rec-result,U_i,(result-data))$\}$ is authentic for $P = SP$ for all ω that contain some evidence for *SP* that U_i has received the result (an action we have not explicitly specified in our example).

  Note that this is **not** non-repudiation of receipt. *SP* knows that the receive action has occurred, but may not be able to proof to anybody else that U_i has received the result.
- A variety of other actions can require authenticity:
  - We may require that *SP* only sends data to users the password of which has been checked, i.e. that whenever *SP* sends data to U_i, the action (rec-login,*SP*,(U_i,username,password)) must be authentic for *SP* itself. This requirement is not relevant for securing the communication between *SP* and U_i but has to be taken into account during the development of the *SP* software.
  - In the context of fair contract signing with a trusted third party (TTP) we may require that the TTP sends the signed contract only if all signers have signed. This enforces authenticity for TTP of all signing actions when sending the signed contract.

3.2 Proof of authenticity

3.2.1 Definition of the property

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, it can send this proof to other agents, which in turn can receive the proof and be convinced of Γ's authenticity. In the following definition the set ΓP denotes actions that provide agents with proofs about the authenticity of Γ. If agent P has executed an action from ΓP then Γ is authentic for P and P can forward the proof to any other agent using actions in ΓS.

Definition 2 (Proof of authenticity) *A pair $(\Gamma S, \Gamma P)$ with $\Gamma S \subseteq \Sigma$ and $\Gamma P \subseteq \Sigma$ 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 $alph(\pi_P(\omega)) \cap \Gamma P \neq \emptyset$ the following holds:*

(1) For P the set Γ is authentic after ω and
(2) for each $R \in \mathbb{P}$ there exist actions $a \in \Sigma_{/P} \cap \Gamma S$ and $b \in \Sigma_{/R} \cap \Gamma P$ 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.

This definition represents one specific type of proofs. Kailar has classified proofs as strong or weak and transferable or non-transferable [8]. 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 its 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 occurred.

3.2.2 Examples

Note that we have not specified actions for forwarding of proofs and receiving of forwarded proofs in our example because in the abstract specification of a system these actions are usually not included and it is simply assumed that general send and receive actions can be performed. Thus these actions are not explicitly included in the properties described below. However, in order to prove that these properties hold, the forwarding and receiving of proofs can easily be added.

- Non-repudiation of origin of (send-request,U_i,(request-data)) by U_i for SP. Before processing the request, the service provider should get a proof showing that user U_i indeed has requested the particular service. Thus we require $P = SP$ to hold a proof of authenticity for $\Gamma = \{$(send-request,U_i,(request-data)) in all sequences ω containing (rec-request,SP, (U_i,request-data)). For any other agent Q the action in Γ shall be authentic after some receive proof action (e.g. some general receive action (receive,Q,(proof)) which we have not included in our example).

- Non-repudiation of receipt of (rec-result,U_i,(result-data)) by U_i for *SP*. The service provider may require a proof that the user has received the result before processing the logoff. In the context of a business process, this can imply that after having received the result the user is obliged to pay the price agreed upon. So we require *SP* to own a proof of the authenticity of $\Gamma = \{$(rec-result,U_i,(result-data)) for all ω that contain an action (rec-logoff,*SP*,(U_i)). On a lower level of abstraction, the precise action that provides the proof to *SP* that (rec-result,U_i,(result-data)) has happened has to be specified.
- In addition to the well-known properties of non-repudiation of receipt and non-repudiation of origin, other requirements can be specified using our notion of proof of authenticity. *SP*, for example, may need to prove to other agents the authenticity of its own send action (send-result,*SP*,(U_i,result-data)). Realisation of this property may require a trusted third party as *SP* must not be able to manipulate the proof.

3.3 Authenticity with respect to a phase

3.3.1 Definition of the property

In many cases it is not only necessary to know *who* has performed a particular action, but also the specific *"time"* of the action is important. As our specification is discrete and does not model any real time properties, time is modeled in terms of relations between actions in a sequence. However, discrete time can be included explicitly by introducing a clock.

We use the definition of a *phase* provided in [6a]. A phase $V \subset \Sigma^*$ is a prefix closed language consisting only of words which, as long as they are not maximal in V, show the same continuation behaviour within V as within S.

Definition 3 *Let $S \subseteq \Sigma^*$ be a system. A prefix closed language $V \subset \Sigma^*$ is a phase in S if the following holds:*

(1) $V \cap \Sigma \neq \emptyset$
(2) $\forall \omega \in S$ *with* $\omega = uv$ *and* $v \in V \setminus (max(V) \cup \{\varepsilon\})$ *holds:* $\omega^{-1}(S) \cap \Sigma = v^{-1}(V) \cap \Sigma$

A phase can be a very complex part of the system. However, often phases have well defined start and end actions. In our example, for *SP* these actions are rec-login and rec-logoff. Therefore, for our purposes it is sufficient to consider only those phases that start with one specific action and end with one specific action. In other words, an element of such a phase begins with the start action and contains either no end action at all or contains just one end action at the end. Authenticity with respect to a phase is the security property provided by authentication protocols.

Definition 4 (Authenticity with respect to a phase) *A set of actions* $\Gamma \subseteq \Sigma$ *is authentic for agent* $P \in \mathbb{P}$ *after a sequence of actions* $x \in S$ *with respect to* W_P *and a phase* V *if it is authentic for* P *after* x *and for all* $y \in \lambda_P^{-1}(\lambda_P(x)) \cap W_P)$ *exists* $u, v, w \in \Sigma^*$ *such that* $y = uvw$ *and* $v \in V$ *and* $alph(v) \cap \Gamma \neq \emptyset$. Γ *is currently authentic for* P *after* x *if* $w = \varepsilon$.

3.3.2 Examples

The service provider *SP* may not only require assurance about *who* has sent a request but *when* the request has been sent. In particular, requests should only be answered when they have been sent during a current session which has been started with a *login* and has not yet terminated. In the example, termination is by explicit *logoff* only. In realistic scenarios, there can be more possibilities, e.g. time-out, error, etc.

Thus we require for all sequences ω in which *SP* has performed an action (rec-request,*SP*,(U_i,request-data)) the authenticity of Γ = {(send-request,U_i,(request-data)) for *SP* with respect to the current phase related to U_i. This phase must have started with (rec-login,SP,(U_i,username,password)) and not yet ended (i.e. (rec-logoff,*SP*,(U_i)) must not yet have occurred for *SP*).

3.4 A remark to integrity

Usually, integrity is viewed as a very important security requirement. It is certainly true that data integrity is a central security requirement. However, consider the following example. Data integrity can be provided by a checksum. Change of the data would be detected because the checksum is not correct any more. But without authenticity, i.e. without knowing who computed the checksum, the checksum mechanism is useless: A malicious agent may change the data and simply compute a new checksum. Integrity is satisfied (meaning that data has not been changed since the checksum has been computed) but the actual security requirement is not.

Thus in order to make sense, integrity of data requires some kind of anchor fixing the origin of the data that shall remain unchanged. When considering distributed systems, this anchor can only be provided by some authentic action in the past. Taking into account that integrity is implied by authenticity and that integrity without authenticity is valueless we have omitted a definition for integrity.

3.5 Confidentiality

3.5.1 Definition of the property

Typically, the well-known concepts of non-interference or information flow control address confidentiality of actions: the occurrence or non-occurrence of certain actions of an agent shall not be deducible for another agent based on what it observes. In the literature there is a variety of formalizations of this concept, Mantel [10] gives a good insight into this topic. The subtle differences between these definitions show the spectrum of this kind of confidentiality.

However, non-interference is not suitable for the specification of security requirements in distributed open systems. Here, it can be assumed that all actions concerned with communication might indeed be visible to malicious agents. Nevertheless, confidentiality is required for data transmitted using these actions. An adequate notion of confidentiality therefore has to provide the flexibility to define confidentiality for arbitrary parameters of the actions. The notion of *parameter-confidentiality* presented in [6c] provides this flexibility.

Various aspects are included in our definition. First, we have to consider agent C's view of the sequence ω it has monitored and thus the set of sequences $\lambda_C^{-1}(\lambda_C(\omega))$ that are, from C's view, identical. Second, C can discard some of the sequences from this set, depending on its knowledge of the system and the system assumptions, all formalized in W_C. There may for example exist interdependencies between parameters in different actions, such as a credit card number remaining the same for a long time, in which case C considers only those sequences of actions possible in which an agent always uses the same credit card number. In the simplified example in this paper, the password is never changed (in contrast to any good security practice). So the set of sequences C considers possible after having monitored ω is reduced to $\lambda_C^{-1}(\lambda_C(\omega)) \cap W_C$. Third, we need to identify the actions in which the respective parameter(s) shall be confidential. Usually many actions are independent from these and do not influence confidentiality, thus need not be considered.

Essentially, in our definition, parameter confidentiality is captured by requiring that for the actions that shall be confidential for agent C with respect to some parameter p, all possible (combinations of) values for p occur in this set. What are the possible combinations of parameters is the fourth aspect that needs to be specified, as we may want to allow C to know some of the interdependencies between parameters (e.g. C may be allowed to know that a message that was received must have been sent before).

Our definition of parameter confidentiality captures all these different aspects. However, due to lack of space we omit the formal details (they can be found, along with the necessary explanations, in [6c]). We will now demonstrate the variability of our definition with various examples.

In order to specify parameter-confidentiality, the following information is needed:

α: the action(s) for which some parameter(s) shall be confidential,
$par(\alpha)$: a mapping par denoting the parameter(s) that shall be confidential
$\mathcal{P}$: the set of possible values for parameter $par(\alpha)$
$\mathbb{Q}$: a set of agents that is allowed to know the parameter value(s)
L: optional, a description of the interdependencies between parameter values.

3.5.2 Examples

An obvious parameter for a confidentiality requirement in our example is the password in the send-login action. Let $\mathcal{P}$ be the set of possible passwords.

For all users U_i, we require $\alpha = \{$(rec-login,SP,(U_i,username,password))$\}$ to be parameter confidential with respect to $par(\alpha)$ = password for all agents P not element of $\mathbb{Q}=\{U_i,SP\}$. We assume that all agents know that agents' passwords are not changed, and that there is an interdependence between sending and receiving, i.e. that the same password is received that was sent. Let us assume that the following sequence of actions has happened:

(send-login,U,(name,pw)) (rec-login,SP,(U,name,pw))

The following examples show that whether or not a certain security property holds in the system depends on agent C's knowledge W_C of the system and on the agent's local view λ_C, as already explained in section 2.1.

- Assume there is a malicious agent C that is able to see send or receive actions of other agents, but not which agent performed the actions nor the data. Thus C's local view of the above action sequence can be defined as follows:
  λ_C((send-login,U,(name,pw)) (rec-login,SP,(U,name,pw))) = send-login rec-login
  Obviously, λ_C^{-1}(send-login rec-login) $\cap\ W_C$ includes all possible passwords for the action rec-login, thus with this local view for agent C the system is parameter confidential as required.
- Let us now assume that for some reason C can not only see the send action of agent U but also the data (name,pw) (for example because U uses a wireless keyboard which can be monitored by C). Since C also knows that SP receives the same data that was sent by U, λ_C^{-1}(λ_C((send-login,U, (name,pw))(rec-login,SP,(U,name,pw))) contains only sequences with (U,

name,pw) as parameter for the receive action. Hence this system is not parameter confidential.

4 Conclusions

We have presented a framework for security requirements specification. A variety of security requirements can be specified within the framework. Examples show that the specification of these requirements is reasonable, simple and understandable. The underlying system model is flexible and can therefore be used for the specification of abstract systems on the business process level, as well as for refined specification of concrete system components, as for example cryptographic protocols. Furthermore, by extending the system specification with the agents' local view and their knowledge about the global system, all parameters needed to reason about the satisfaction of security properties are included into the system specification. We believe that all this has to be taken into account in the development of secure systems.

Precise specification of security requirements is a necessary basis for the development of secure applications. However, the usability of the presented framework goes beyond requirements specification, as it can serve as the basis of rigorous treatment of security throughout the complete development process.

References

[1] M. Abadi and M.R Tuttle. A Semantics for a Logic of Authentication. In *Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Canada*, pages 201–216, August 1991.

[2] D. Bell and L. La Padula. Secure computer systems: Unified exposition and multics interpretation. Technical report MTR-2997, Mitre Cooperation, 1975.

[3] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8, 1990.

[4] CASENET. IST project 2001-32446. http://www.casenet-eu.org/.

[5] D. Clark and D. Wilson. A comparison of commercial and military computer security policies. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1987.

[6a] R. Grimm and P. Ochsenschläger. Binding Cooperation, A Formal Model for Electronic Commerce. *Computer Networks*, 37:171–193, 2001.

[6b] S. Gürgens, P. Ochsenschläger, and C. Rudolph. Authenticity and provability - a formal framework. In *Infrastructure Security Conference InfraSec 2002*, volume 2437 of *Lecture Notes in Computer Science*, pages 227–245. Springer Verlag, 2002.

[6c] S. Gürgens, P. Ochsenschläger, and C. Rudolph. Parameter confidentiality. In *Informatik 2003 - Teiltagung Sicherheit*. Gesellschaft für Informatik, 2003.

[7] M. Heisel, Pfitzmann A., and Santen T. Confidentiality-preserving refinement. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 295–305. IEEE Computer Society Press, 2001.

[8] R. Kailar. Accountability in Electronic Commerce Protocols. *IEEE Transactions on Software Engineering*, 22(5):313–328, 1996.

[9] V. Lotz. Threat Scenarios as a Means to Formally Develop Secure Systems. *Journal of Computer Security 5*, pages 31 – 67, 1997.

[10] H. Mantel. Possibilistic definitions of security – an assembly kit. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 185–199, 2000.

[11] J. McLean. The specification and modeling of computer security. *IEEE Computer*, 23(1):9–16, 1990.

[12] L. C. Paulson. Proving Properties of Security Protocols by Induction. Technical Report 409, Computer Laboratory, University of Cambridg, 1996.

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

[14] G. Wedel and V. Kessler. Formal Semantics for Authentication Logics. In *Computer Security - Esorics 96*, volume 1146 of *LNCS*, pages 219–241, 1996.

[15] A. Zakinthinos and E. Lee. A general theory of security properties. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1997.