IPSJ Journal Vol. 52 No. 7 2108–2125 (July 2011)

Regular Paper

Formal Notions of Trust and Confidentiality–Enabling Reasoning about System Security

Andreas Fuchs,[†1] Sigrid Gürgens[†1] and Carsten Rudolph[†1]

Historically, various different notions of trust can be found, each addressing particular aspects of ICT systems, e.g., trust in electronic commerce systems based on reputation and recommendation, or trust in public key infrastructures. While these notions support the understanding of trust establishment and degrees of trustworthiness in their respective application domains, they are insufficient when addressing the more general notion of trust needed when reasoning about security in ICT systems. Furthermore, their purpose is not to elaborate on the security mechanisms used to substantiate trust assumptions and thus they do not support reasoning about security in ICT systems. In this paper, a formal notion of trust is presented that expresses trust requirements from the view of different entities involved in the system and that enables to relate, in a step-by-step process, high level security requirements to those trust assumptions that cannot be further substantiated by security mechanisms, thus supporting formal reasoning about system security properties. Integrated in the Security Modeling Framework SeMF this formal definition of trust can support security engineering processes and formal validation and verification by enabling reasoning about security properties with respect to trust.

1. Introduction

The meaning of the term *trust* in the context of information and communication technology (ICT) systems differs from the concept of trust between people. In particular trust as seen in the notion of *trusted computing* (e.g., as defined by the Trusted Computing Group TCG) refers to particular properties of a technical system. This notion of trust stands in contrast to some more intuitive notions of trust expressing that someone behaves in a particular well-behaved way. Trust in a technical system always has to be seen as trust in a property of the system. A more meta-level generic trust as is possible for people ("I trust you") is not useful for computers or technical entities as parts of communication networks. A variety of existing notions of trust in the context of ICT systems addresses particular aspects, e.g., trust in electronic commerce systems based on reputation and recommendation, or trust in public key infrastructures (see Section 3 for a survey). While these notions are useful for understanding trust establishment and degrees of trustworthiness in these application domains, they cannot be used for a more general notion of trust needed to reason about trust in ICT systems. In addition to the restricted applicability there is also a lack of formal semantics for the properties expressed by these different notions of trust. However, when used in a security engineering process, formal semantics are essential for traceability of trust and security requirements through the different steps of the process. This traceability is necessary to show relations between high-level requirements and underlying security mechanisms (e.g., particular cryptographic algorithms) and trust assumptions (e.g., trust in hardware security or trust in a particular behaviour of people using the system).

The goal of the formal notion of trust presented in this paper is to be able to exactly express trust requirements for ICT systems from the view of the different entities involved in the system, and to support formal reasoning such that finally security requirements, security and trust mechanisms and underlying trust assumptions can be formally linked and made explicit. It must be possible to refine complex or implicit trust assumptions (e.g., trust in the "well behaviour" of a certification authority) into basic trust assumptions that can more easily be verified when assessing the security of a concrete ICT system. Hence, such a formal notion of trust can support security engineering processes as well as formal validation and verification. Previously established notions for security properties with formal semantics can provide traceability in a security engineering process as well and are used for validation and verification. However, trust adds another layer of information. While security properties may or may not be global properties of the system, trust always expresses the view of a particular entity or agent of the system. Trust depends on the individual perception of the agents. Therefore, different agents can trust in properties that may be contradictory. Furthermore, it must also be possible to express that one agent trusts that another agent has trust in a particular property (e.g., for expressing trust in

†1 Fraunhofer Institute for Secure Information Technology SIT

certification authorities). It seems that explicit description of trust increases the complexity of reasoning about security by adding another layer of information. However, it should be noted that the information included in trust assertions and trust assumptions is essential for a proper security assessment and risk analysis process. In fact, this layer of information is usually at least partly covered or implicitly included. Explicit and exact representations can help to get a clearer view on the trust assumptions that a risk analysis is based on and thus improve the technical processes as well as the perception of the results of a risk analysis.

The notion of trust presented here extends the existing security modelling framework SeMF[1]. This framework uses formal languages and is independent of specific representations of the system. The example used throughout the paper discusses trust of an agent in that specific authenticity and confidentiality properties hold in a system. A preliminary version has shown the applicability of the new notion of trust for authenticity properties[2]. However, particularly challenging is the handling of confidentiality properties. A large variety of trust assumptions is necessary for assurance in satisfaction of a confidentiality property. Our example is used to explain the new notion of trust in the context of authenticity and confidentiality properties and to show how reasoning can lead to refined trust properties that express underlying trust assumptions and assumptions on security mechanisms. These trust assumptions can subsequently be further refined or the risk of violation can be part of risk analysis processes without further refinement.

The following two sections first provide our terminology and then briefly discuss its relation to existing notions of trust. In Section 4 an example is introduced that will be used throughout the rest of the paper and that imposes some interesting questions related to trust. Section 5 then gives a summary of our Security Modeling Framework SeMF and based on this explains the formal notion of trust. A number of implications and assumptions provided in Section 6 are then used in Section 7 to formally prove some security properties for a formally specified version of the example.

2. Terminology

The meaning of the word *trust* has been subject to many (more or less philosophical) discussions, many different interpretations with subtle differences exist. Achieving a common understanding of the term trust is further complicated by mixing it with the related notions of *trustworthiness* or *reputation*. The formal notion of trust introduced in this paper is supposed to be useful mainly for reasoning about trust in the context of technical systems in the area of ICT. The work was motivated by concepts such as trusted computing using the so-called Trusted Platform Module (TPM)[3]. We do not intend to contribute to the philosophical discussions on trust or the relation between trust and reputation.

Within this paper we will use the following terminologies:
The term ***trust*** refers to a relation from one agent in the system to another agent with respect to a property, or from an agent directly to a property in the system. Thus, agents can have three slightly different types of trust:

(1) Agents can trust that some (global) property holds in a system.
(2) Agents can trust that another agent behaves in a certain way, i.e., that a property concerning the behaviour of this other agent is satisfied.
(3) Agents can trust that another agent has a particular trust.

Being a relation, this notion of trust cannot be used to express different degrees of trust. Agents can either trust or not trust. In a refinement process the notion of trust can be broken down into more detailed trust assumptions. These are expressed using the same formal notion of trust. However, as input for a subsequent security evaluation or risk assessment it is necessary to express to what degree this trust can be substantiated, i.e., what is the *trustworthiness*. Thus, we clearly distinguish between trust and trustworthiness. This motivates the following notion of trustworthiness.

The term ***trustworthiness*** expresses the degree to which a particular trust assumption can be made. Trustworthiness can be expressed as a probability or can simply have fixed values (e.g., high, medium, low). Depending on the particular representation of trustworthiness, agents within the system can reason about the *trustworthiness* of other agents, or reasoning mechanisms can be used for risk analysis and risk assessment.

3. Related Work

A huge part of approaches that use a notion of trust is concerned with reputation systems. In this area, trust is understood in the sense of trustworthiness as explained in Section 2 and e.g., defined by the research project Trust4All[4]:

"Trust is the degree to which a trustor has a justifiable belief that the trustee will provide the expected function or service."

Jøsang, et al.[5] present two different notions of trust that capture main aspects in the context of reputation systems:

- **Reliability Trust**: "Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends."
  This definition captures both the concept of dependence on the trusted party and the non-binary nature of trust in the context of reputation systems.
- **Decision Trust**: "Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible. (inspired by Ref. 6))"
  Hence the concept of Decision Trust is useful in the context of risk assessment.

This or similar characterizations of "trust" can be found in many of the approaches in this area. A very good overview of approaches regarding trust in reputation systems along with a clarification and classification of the main concepts is given by Jøsang, et al.[5]. They explain these concepts further by means of temporary reputation based systems such as eBay. Another survey that focuses particularly on trust management systems is given by Grandison and Sloman[7].

A second branch of research focuses on formal models that capture certain aspects of trust. In Ref. 8) Carbone, et al. introduce a formal model of trust that focuses on the formation, evolution and propagation of trust. Trust in their model is viewed as a function that assigns values to pairs of principles. The model can be used to formulate trust policies, however these seem to be restricted to access control. Further, their approach is not aimed at reasoning about security properties holding or not holding in a system. Demolombe[9] provides a formal definition for trust that distinguishes between different properties an agent may have trust in. Axioms related to these properties are defined, and the resulting axiomatic structure can then be used to reason about conditional trust between agents with respect to ratings regarding aspects such as cooperativity and credibility.

Another branch of research that takes a similar axiomatic approach are the so-called authentication logics. These logics use a specific notion of trust and are aimed at reasoning about security properties of a system. In particular, these logics are useful for the security verification of cryptographic protocols. The first such logic was the BAN Logic[10] by Burrows, et al. Here the concept of *jurisdiction* models trust of agents in statements of specific other agents about for example the trustworthiness of a key. The BAN logic inspired a large number of similar logics (see for example Refs. 11)–14)). Each of these logics constitutes an axiomatic system, i.e., formulates axioms and inference rules that capture the nature of security mechanisms and proves that certain security properties are provided given that certain assumptions on the system hold.

Although these approaches seem to be closely related to the one introduced in this paper, there is a fundamental difference: Our formalization of trust and the thereby enabled reasoning about security properties does not use *axioms* but is *based on a formal semantics* that uses only formal language theory. Further, our notion of trust applies to any security property and is independent of any security mechanism that might be employed to achieve a security property, while the security properties handled by authentication logics are directly derived from specific aspects of security mechanisms.

As already explained in Section 1, we do not address trust in the context of reputation systems. Our notion of trust refers to particular properties of a technical system. Our work aims at providing means to support formal reasoning such that security requirements, security and trust mechanisms, and underlying trust assumptions can be formally linked and made explicit. However, reputation systems can be seen as complementary to our approach. Results achieved by our reasoning can be used as input for a subsequent security evaluation or reputation based risk assessment where it is necessary to express to which degree trust can be substantiated. On the other hand, reputation systems can be used for substantiating trust assumptions being input for our reasoning.

One focus of the example in this paper is on trust assertions related to cryptographic keys. In the literature on Public Key Infrastructures (PKIs) trust and underlying implicit assumptions is a major topic. Ellison and Schneider[15] list as the number one risk in PKIs the question "Who do we trust, and for what?". Although they focus on the questionability of trust in certificate authorities (CA), it also applies to the question of what kind of statement the CA expresses when issuing a certificate.

Ozols, et al. also account for the meaning of certificates and the necessity to formalize trust, "The notion of trust is fundamental in PKIs"[16]. They introduce a state-based model for PKIs "in which a PKI is formally specified by its certification topology and PKI states". However, this formalization focuses on the certification path to the end user rather than on the trust assurances the end user may derive from it and their implications for the follow-up security mechanisms. Huang and Nicol[17] developed a calculus for PKIs and identity management that also targets the certification path and utilizes the reputational understanding of trust to reason about the relative risk when trusting a digital certificate. However their focus was not the inter-operation of the PKI's trust assurance and its implication for the follow-up cryptographic mechanisms.

Regarding the interpretation of digital certificates in PKIs, all of these approaches directly reason about the binding between the public key and a specific agent. However, in this work we will use a different interpretation that allows for an understanding closer related to the assumptions inherent in cryptographic mechanisms.

The main goal of the work presented in this paper is to make explicit the basic assumptions that need to hold for a system by constructing formal relations between security properties, trust requirements and assumptions. This work shall also provide the link between formal security models representing dynamic processes on the one hand and organisational (mostly static) models representing static trust relations as for example expressible by the Secure TROPOS method[18] on the other hand.

4. An Example

In this section we introduce a very simple use case that includes security requirements involving specific trust requirements. Similar situations typically arise in many scenarios from different domains, such as car-to-car, distributed sensor networks or email. This use case serves both as a motivation for the concept of trust and as an example of how to use this concept in order to prove that specific security mechanisms together with certain trust assumptions result in the satisfaction of specific security properties.

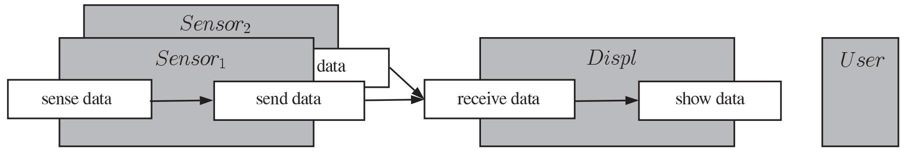

Fig. 1 Example system.

However, this section discusses the security properties and resulting trust requirements only informally in order to give an understanding of the practical implications of our notion of trust. Subsequent sections provide a brief introduction to the formal Security Modeling Framework (SeMF), introduce the formal definition of trust, some resulting theorems, and revisit the example formally. Our scenario is illustrated in **Fig. 1**.

The example system consists of three active nodes and an end user. $Sensor_1$ and $Sensor_2$ are nodes deployed somewhere in the system. They perform measurements (e.g., measure the temperature inside and outside a house) and send the resulting data over the network. *Displ* is the third node of the system. It receives data from the network and displays them to the end user *User* (e.g., the owner of the house).

An obvious requirement of the above use case is that the user, when seeing some data on the display, e.g., the temperature outside the house, wants this data to be indeed measured by the respective sensor. This requirement is usually denoted by *data origin authenticity* and can be informally stated as follows:

P0 *It must be authentic for the end user that the data he/she is shown on the display is the data that was measured by the respective sensor.*

There exist several schemes that can be used to secure a communication channel and provide data origin authenticity for a message during transfer over the network (between *send* and *recv* in Fig. 1), such as digital signature schemes or message authentication codes (MACs). Whatever mechanism is used, the user, a human being, cannot validate a digital signature or MAC. This validation has to be done by the display node. Therefore it is only the display node that can be assured of the authenticity and not the user.

Accordingly what can actually be provided when applying these mechanisms to our use case is that each time the display node receives some data and verifies the signature or MAC with a cryptographic key, it can be sure that the signature or MAC was generated using the corresponding counterpart. However, the mechanism by itself does not provide any means for the display to determine which agent actually performed the signature. A prerequisite for providing data origin authenticity is the confidentiality of the private signature or MAC key. This global property by itself is not sufficient, we have the additional requirement that the display needs to trust that this property indeed holds in the system. Since usually the display has no global view of the system there needs to be some other basis for assurance regarding this property. Without this assurance, even if confidentiality of the private or MAC key globally holds in the system, the display cannot consider this requirement satisfied and will act as if the key might be known to others. In practice Public Key Infrastructures and corresponding digital certificates are used as a means for this kind of trust assertions from one entity to another.

Yet, even this assurance does not extend the origin authenticity to the action of measuring the data. The sensor can very well sign data different to the one that it has measured. Also, it is the action of showing the data that is relevant for the user, not the action in which the display node receives and verifies it (the display might show data different to the one received). Finally, since it is the user who wants the data to be measured, it is the user who needs to trust that all these properties hold.

In order to capture this situation and simplify the example, we abstract the sensor actions of signing and sending the data to a sending action, the display actions of receiving and verifying the data to a receiving action, but keep the measuring and displaying actions, respectively, separate. The discrepancy between property P0 we want the system to provide and the property that can be provided by a digital signature or MAC is illustrated in **Fig. 2**.

For the example we assume that a secure signature scheme is used and that the necessary trust into the confidentiality of signature keys is established by way of a PKI structure. We therefore assume that the system manufacturer in the role of a Trusted Third Party (TTP) provides the display with the sensors' certificates

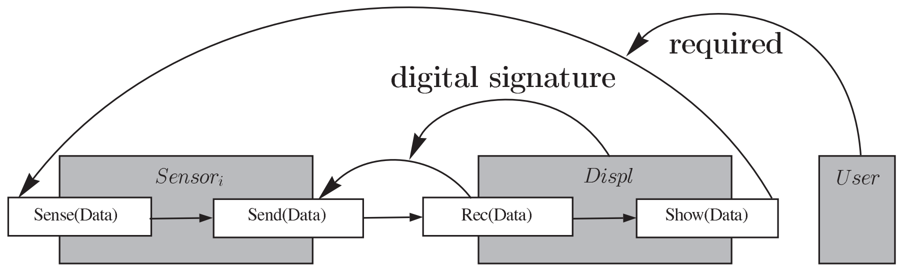

Fig. 2 Discrepancy between required property and signature mechanisms.

for their respective public keys that express the possession of the corresponding private key as well as the trustworthiness of the owners' behavior.

The use of a PKI in order to satisfy these trust assertions leads to a more explicit interpretation of the purpose of digital certificates, contributing to formal analysis of the overall system's security. Whereas usually a certificate's meaning is interpreted as binding a public key to an agent, we now express that a certificate's meaning is to phrase an assurance about the confidentiality of the public key's private counterpart. If the display trusts the TTP with respect to statements about the confidentiality of keys and well-behavior of key owners, and if the display owns a certificate issued by this TTP referring to a sensor's public and private key, then the display may trust in that the sensor's private key is indeed confidential to the sensor and that the sensor behaves as expected. This interpretation of a certificate is actually the idea behind the BAN logic 'jurisdiction rule". However, when applying BAN logic, a complex trust assumption (trust in the jurisdiction of the TTP with respect to confidentiality of a key) has to be verified. In contrast, our approach allows to refine the trust in the confidentiality of the sensor's private key and the sensor's behavior to trust into e.g., the confidentiality of the TTP's private key, the correct working of the TTP's hardware and software, etc. However, for lack of space we do not discuss this refinement process further. Note that at some point in a practical process it is necessary to end the formal reasoning and switch to risk analysis processes.

Our notion of trust introduced in Ref. 2) allows to formalize this type of implications and can also serve as a formal and exact basis for risk analysis.

An agent trusts in a property to hold in a system if in *its conception* of the system this property is fulfilled.

In the next section we give a brief introduction to our Security Modeling Framework SeMF and give the formal definitions of those security properties that will be used in this paper to demonstrate our notion of trust.

5. The Security Modeling Framework SeMF

The behaviour B of a discrete system S can be formally described by the set of its possible sequences of actions (traces). Therefore $B \subseteq \Sigma^*$ holds, here Σ (called the alphabet) is the set of all actions of the system, Σ^* is the set of all finite sequences (called words) of elements of Σ, including the empty sequence denoted by ε, and subsets of Σ^* are called formal languages. Words can be composed: if u and v are words, then uv is also a word. For a word $x \in \Sigma^*$, we denote the set of actions of x by $alph(x)$. For more details on the theory of formal languages we refer the reader to Ref. 19).

We further extend the system specification by two components: *agents' initial knowledges* about the global system behaviour and *agents' local views*. The initial knowledge $W_P \subseteq \Sigma^*$ of agent P about the system consists of all traces P initially considers possible, i.e., all traces that do not violate any of P's assumptions about the system. Every trace that is not explicitly forbidden can happen in the system. An agent P may assume for example that a message that was received must have been sent before. Thus the agent's W_P will contain only those sequences of actions in which a message is first sent and then received. Further we can assume $B \subseteq W_P$, as reasoning within SeMF primarily targets the validation and verification of security properties in terms of positive formulations, i.e., assurances the agents of the system may have. Other approaches that deal with malfunction, misassumptions and attacker models cannot rely on this assumption.

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 $\omega \in B$ has happened, every agent P can use its *local view* λ_P of ω to determine the sequences of actions it considers to have possibly happened. Examples of an agent's local view are that an agent can see only its own actions, or that an agent P can see the message that was sent over a network bus but cannot see who sent it, in which case e.g., $\lambda_P(send(sender, message)) = send(message)$.

For a sequence of actions $\omega \in B$ and agent $P \in \mathbb{P}$ ($\mathbb{P}$ denoting the set of all agents), $\lambda_P^{-1}(\lambda_P(\omega)) \subseteq \Sigma^*$ is the set of all sequences that look exactly the same as ω from P's local view. In the above case in which P cannot see the sender of a message, sequences for example that contain an action from $\{send(sender_1, message), send(sender_2, message), \ldots\}$ all look the same for P. 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 initial knowledge to reduce this set: $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$ describes all sequences of actions P considers to have possibly happened when ω has happened.

Security properties can now be defined in terms of the agents' initial knowledges and local views. In Ref. 1) we have introduced a variety of definitions of security properties (e.g., authenticity, proof of authenticity, confidentiality). Our concept of trust introduced in Ref. 2) applies to all of them, however, we will use our notions of authenticity and parameter confidentiality as a demonstrating example.

We call a particular action a authentic for an agent P (after a sequence of actions ω has happened) if in all sequences that P considers to have possibly happened, a must have happened (some time in the past). 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 authentically 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.

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

We define the following instantiation of this property that states that whenever an action b has happened in a sequence of actions ω, it must be authentic for agent P that action a has happened as well. Note that in most cases, action b is in P's local view.

Definition 2 For a system S with behaviour $B \subseteq \Sigma^*$, agent $P \subseteq \mathbb{P}$, and actions $a, b \in \Sigma$, $auth(a, b, P)$ holds in B if for all $\omega \in B$, whenever $b \in alph(\omega)$, the action a is authentic for P.

The precedence of actions is a weaker property:

Definition 3 For a system S with behaviour $B \subseteq \Sigma^*$ and sets of actions $\Sigma_1, \Sigma_2 \subseteq \Sigma$, $precede(\Sigma_1, \Sigma_2)$ holds in S if for all $\omega \in B$ with $\Sigma_2 \cap alph(\omega) \neq \emptyset$ it follows that $\Sigma_1 \cap alph(\omega) \neq \emptyset$.

In the following we give a brief introduction and the formal definition of *parameter confidentiality* as introduced in Ref. 20).

Parameter Confidentiality essentially captures the following: Assume an agent R has monitored a sequence of actions ω, and in some of these actions a parameter p occurs with a specific value. Then R must not be able to distinguish this sequence from any other sequence in which the parameter occurs with different values, even if knowing all possible values.

Considering the example introduced in Section 4, we may want to require that a sensor's signature key shall be confidential to all other agents. Now assume that $Sensor_1$ generates and sends a signature and *Displ* receives, verifies and accepts this signature, modeled for example by $send(Sensor_1, data, privK_1, sig_j(data))\ recv(Displ, data, pubK_1, sig_j(data))$. When monitoring this sequence, $Sensor_2$ and the display shall not be able to deduce the actual private key that was used even if knowing the key's length and thus all possible values. This property can be expressed with our notion of parameter confidentiality and we will explain it further using the requirement of $privK_1$ to be confidential to the display.

Various aspects are included in this definition. First, one has to consider *Displ*'s view of the sequence ω it has monitored and thus the set $\lambda^{-1}_{Displ}(\lambda_{Displ}(\omega))$ of sequences that are, from *Displ*'s view, identical. We assume use of a network bus connection that allows all agents to see their own actions and the data and signature of actions performed by other agents, but not the sender and the signature key. The display's local view λ_{Displ} of the sequence above is then $\lambda_{Displ}(send(Sensor_1, data, privK_1, sig_j(data))\ recv(Displ, data, pubK_1, sig_j(data)) = send(data, sig_j)\ recv(Displ, data, pubK_1, sig_j(data))$. The set $\lambda^{-1}_{Displ}(\lambda_{Displ}(send(Sensor_1, data, privK_1, sig_j(data))\ recv(Displ, data, pubK_1, sig_j(data))))$ of sequences of actions that are from the display's view identical consists of sequences $send(P, data, privK_i, sig_j(data'))\ recv(Displ, data, pubK_1, sig_j(data))$ with $privK_i$ denoting possible key values, P denoting any possible sender, and $data'$ denoting possible values of the parameter $data$ in $sig_j(data')$.

Second, *Displ* can discard some of the sequences from this set, depending on its knowledge of the system and the system assumptions, all formalized in W_{Displ}. There may for example exist interdependencies between parameters in different actions, such as the public key used for a signature's verification determining the private key used for its generation. In consequence, *Displ* considers only those sequences of actions possible in which its own receive action, including verification and acceptance of the signature, is always preceded by a signature generation and send action which uses the same data and the signature key corresponding to the public key used for verification. So the set of sequences *Displ* considers to have possibly happened after ω has happened is reduced to $\lambda^{-1}_{Displ}(\lambda_{Displ}(\omega)) \cap W_{Displ}$. In the example this set consists of sequences $send(P, data, privK_i, sig_j(data))\ recv(Displ, data, pubK_1, sig_j(data))$, with *Displ* knowing that there is a relation between the public key $pubK_1$ used for verification of the signature and $privK_i$ but not being able to deduce the actual value of $privK_i$.

Third, those actions have to be identified in which the respective parameter(s) shall be confidential, or in other words, the actions from which a monitoring agent can possibly learn the parameter's value. Usually many actions are independent from these and do not influence confidentiality and thus need not be considered. In the example, it is the send action that contains $privK_i$ and from which *Displ* can learn its value but the sensing and display actions are not relevant regarding the confidentiality of $Sensor_1$'s signature key. We formalize this by using a homomorphism μ that maps all actions of interest for the particular confidentiality property onto what we call the *action type* while at the same time extracting the parameter to be confidential, and that maps all other actions onto the empty word. In the example, we define $\mu(send(Sensor_i, data, privK_i, sig_j(data))) = (send(Sensor_i, data, sig_i), privK_i)$

and $\mu(action) = \varepsilon$ for all other actions *action*.

Essentially, parameter confidentiality is captured by requiring that for actions that shall be confidential for an agent with respect to some parameter p, all possible (combinations of) values for p occur. What are the possible combinations of parameters is the fourth aspect that needs to be specified, as one may want to allow the agent to know some of the interdependencies between parameters (e.g., it may be allowed to know the relation between signature and verification key). The notion of (L, M)–Completeness captures which are the allowed dependencies within a set of sequences of actions.

L is a formal language that consists of sequences of pairs (*action type, number*) where *action type* are the types of actions that are kept by μ and the numbers are used to identify those actions whose relation between the parameters is allowed to be known. We extend the example and add a further send action by $Sensor_1$ and the respective receive action by $Displ$, so $\omega = send(Sensor_1, data, privK_1, sig_i(data)) \quad send(Sensor_1, data, privK_1, sig_j(data))$ $recv(Displ, data, pubK_1, sig_i(data)) recv(Displ, data, pubK_1, sig_j(data))$.

Now while $Displ$, owning the sensor's public key and thus knowing (and being allowed to know) that $Sensor_1$ always uses the same private key, $Sensor_2$ does not own $pubK_1$ and thus should not be able to distinguish one send action from another. Hence for describing the confidentiality requirement regarding the display we assign all send actions the same number, hence L_{Displ} contains sequences $(send(Sensor_1, data, sig_i), k)(send(Sensor_1, data, sig_j), k)$. Addressing the confidentiality requirement of $Sensor_2$ we assign different numbers to the send actions which results in L_{Sensor_2} containing sequences $(send(Sensor_x, data, sig_i), k)(send(Sensor_x, data, sig_j), l)$. Functions f are then used to map these numbers to the set M of possible parameter values. The resulting set of action sequences contains all possible combinations of parameter values with the constraint that actions related with respect to the parameter contain the same parameter value. Such a set of action sequences is called (L, M)–complete.

For the formal definition of (L, M)–completeness, some additional notation is needed: For $f : M \longrightarrow M'$ and $g : N \longrightarrow N'$ we define $(f, g) : M \times N \longrightarrow M' \times N'$ by $(f, g)(x, y) := (f(x), g(y))$. The identity on M is denoted by $i_M : M \longrightarrow M$, while $M^{\mathbb{N}}$ denotes the set of all mappings from $\mathbb{N}$ to M.

Definition 4 Let $L \subseteq (\Sigma_t \times \mathbb{N})^*$ and let M be a set of parameters. A language $K \subseteq (\Sigma_t \times M)^*$ is called (L, M)–complete if

$$K = \bigcup_{f \in M^{\mathbb{N}}} (i_{\Sigma_t}, f)(L)$$

The definition of parameter confidentiality captures all the different aspects described above:

Definition 5 (Parameter Confidentiality) Let M be a parameter set, Σ a set of actions, Σ_t a set of types, $\mu : \Sigma^* \to (\Sigma_t \times M)^*$ a homomorphism, and $L \subseteq (\Sigma_t \times \mathbb{N})^*$. Then M is *parameter confidential* for agent $R \in \mathbb{P}$ with respect to (L, M)–completeness if there exists an (L, M)–complete language $K \subseteq (\Sigma_t \times M)^*$ with $K \supseteq \mu(W_R)$ such that for each $\omega \in B$ holds

$$\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R) \supseteq p_1^{-1}(p_1(\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R))) \cap K$$

Here $p_1^{-1} \circ p_1$ first removes and then adds again the parameters that shall be confidential, i.e., constructs all possible value combinations. (L, M)–completeness of K captures exactly that we require R to consider all combinations of parameter values possible except for those that we allow to be disregarded. Hence the right hand side of the inequality constitutes all sequences of actions we require R to consider to have possibly happened after ω has happened, while the left hand side constitutes those sequences R actually does consider to have possibly happened. For further explanations we refer the reader to Refs. 20), 21).

The predicate *conf* denotes a particular instantiation of parameter confidentiality. In the following definition, *par* denotes the parameter that shall be confidential, $\mathcal{A}(par)$ denotes the actions that contain *par* and can extend agents' knowledge about the parameter values, i.e., the actions corresponding to those identified by μ. For simplicity we fix the set M of values for the parameter *par*. Further, as our example does not consider privacy or anonymity features, we allow agents not in the set *who* to know all interdependencies between actions in $\mathcal{A}(par)$ (denoted by L_{max}). Then this predicate expresses that all agents not being element of *who* shall not be able to distinguish between values of *par*.

Definition 6 (confidential) Let $who \subseteq \mathbb{P}$ be a subset of agents, *par* be the parameter whose value shall only be known by the agents in *who*, and $\mathcal{A}(par)$ be the set of actions from which a malicious agent can extract knowledge about the value of *par*. Then $conf(\mathcal{A}(par), par, who)$ holds in B if *par* is parameter-confidential for all $P \in (\mathbb{P} \setminus who)$ with respect to (L_{max}, M)–completeness according to Definition 5.

In the following, the definition of a system includes all factors that are important regarding security properties holding or not holding in a system.

Definition 7 (System) A system $S = (\Sigma, \mathbb{P}, B, \mathbb{W}, \mathbb{V})$ consists of a set $\mathbb{P}$ of agents acting in the system, a language $B \subseteq \Sigma^*$ over an alphabet of actions Σ describing the system behaviour in terms of sequences of actions, a set $\mathbb{V} = \{\lambda_X : \Sigma^* \rightarrow (\Sigma_X)^* | X \in \mathbb{P}\}$ of agents' local views, and a set $\mathbb{W} = \{W_X \subseteq \Sigma^* | X \in \mathbb{P}\}$ of agents' initial knowledges.

Here $(\Sigma_X)^*$ denotes the image of the homomorphism λ_X which has to be individually specified for each system. Which part of an action an agent can see depends on the specific system to specify and can contain any part of it, as indicated in the previous section.

An agent P's conception and understanding of a system S, denoted by S_P, may differ from the actual system. P may not know all about the system's behaviour, thus from P's point of view the system's behaviour consists of P's initial knowledge W_P. Further, P may not have all information with respect to the other agents' initial knowledges and local views, so P's conception of agents' initial knowledges $(W_{X\,P})$ and local views $(\lambda_{X\,P})$ may differ from the actual initial knowledges and local views of the system S. This motivates the following definition.

Definition 8 (Trusted System) Agent P's conception of system S is defined by $S_P = (\Sigma, \mathbb{P}, W_P, \mathbb{W}_P, \mathbb{V}_P)$. Σ and $\mathbb{P}$ are the alphabet and set of agents, respectively, of both S and S_P, whereas P's initial knowledge (conception) $W_P \subseteq \Sigma^*$ of system behaviour B constitutes the behaviour of S_P. It further contains a set $\mathbb{V}_P = \{\lambda_{X\,P} : \Sigma^* \rightarrow (\Sigma_{X\,P})^* | X \in \mathbb{P}\}$ of agent P's conception of agents' local views of S, and a set $\mathbb{W}_P = \{W_{X\,P} \subseteq \Sigma^* | X \in \mathbb{P}\}$ of agent P's conception of agents' initial knowledges in S. We say that P *trusts* in system S_P (since it represents P's knowledge about system S).

The definition of an agent's trusted system gives rise now to the definition of an agent's *trust in a property* holding in a system:

Definition 9 (Trusted Property) Let *prop* be any property that refers to a system as defined in Definition 7. An agent $P \in \mathbb{P}$ trusts in *prop* to hold in a system S, denoted by *trust(P, prop)*, if *prop* is fulfilled in S_P.

This notion of trust follows naturally from the different aspects that constitute the model of a system. If a property holds in the system as P perceives it (i.e., in S_P), then from P's point of view the property holds, i.e., P trusts in the property to hold in S. Further the notion of trust allows to specify precisely what it is an agent trusts in. An agent may have trust in one property but not in another. Of course, trust itself is a property of a system as well. Therefore the trust concept allows to model arbitrarily long trust chains such that e.g., the trust of an agent in another agent's trust in a property can be expressed.

6. Implications between Trust Properties Given Sufficient Assumptions

This section presents sufficient assumptions and proves specific implications of security and trust properties of SeMF that will then be used to reason about properties provided by the example system when introducing certain security mechanisms. A more detailed analysis of the presented sufficient assumptions is given in Section 7.2.

6.1 Sufficient Assumptions When Reasoning about Trust

The very first assumption in SeMF (see Section 5) is that no agent falsely excludes behaviour that can actually happen in the system, i.e., that no agent has false knowledge. As SeMF is concerned with assurance-based security reasoning, an agent only takes those facts for granted that can safely be assumed to hold, hence this assumption seems reasonable.

Since this assumption holds for all systems, it holds in particular for a trusted system. The interpretation of the respective formal statement reveals more insight into the relationship between agents' initial knowledges: An agent cannot assign another agent more knowledge than it has itself. Note that the more an agent knows, the smaller its initial knowledge. Formally, this assumption can be expressed as follows:

Assumption 1 In general the behaviour of a system is included in all agents' initial knowledge (see Section 5). Hence for all $P, Q, Q', \ldots \in \mathbb{P}$, $B \subseteq W_P$, $W_P \subseteq W_{Q\,P}$, $W_{Q\,P} \subseteq W_{Q'\,Q\,P}$, etc.

We further assume that agent P does not assign agent R a more precise initial knowledge than this agent R actually has. This relation reflects that if P falsely assumed the satisfaction of properties for R then P would draw the wrong conclusions regarding its own trusted properties:

Assumption 2 The initial knowledges of agents $P, R, Q, Q', \ldots \in \mathbb{P}$ in the systems S and S_P, S_P and S_{Q_P}, S_{Q_P} and $S_{Q'_{Q_P}}$, etc., are related to each other in the following way: $W_R \subseteq W_{RP}$, $W_{RP} \subseteq W_{RQ_P}$, $W_{RQ_P} \subseteq W_{RQ'_{Q_P}}$, etc.

For local views of agents we have the analogous relation: What agent P assumes regarding agent R's local view is not more concrete than R's actual local view:

Assumption 3 For agents $P, R, Q, Q', \ldots \in \mathbb{P}$ we assume the existence of a homomorphism h_{RP} with $\lambda_{RP} = h_{RP} \circ \lambda_R$ that maps R's local view onto P's conception of R's local view. Assuming its existence simply formalizes that the former is fuzzier than the latter. Similarly we assume the existence of such homomorphisms for arbitrary depths of nested trusted systems (e.g., that exists a homomorphism h_{RQ_P} with $\lambda_{RQ_P} = h_{RQ_P} \circ \lambda_{RP}$, etc.).

The final assumption addresses particular aspects of parameter confidentiality. Its first part of the conjunction describes agent P's view of those actions that extend R's knowledge about the parameter to be confidential (i.e., those that are relevant for the confidentiality of a particular parameter identified by the homomorphism μ) must match the actions that actually extend R's knowledge, in other words P must be able to identify all actions that add to R's knowledge. These actions can represent regular system behaviour or attacks such as zero-day-exploits, unknown "covert channels" or weak/predictable RNGs. The second part of the assumption describes that agent P must not assume that R sees less than it actually does. Wrong assumptions on R's local view can directly lead to wrong conclusions on confidentiality properties. If R is able to see more actions (e.g., through side-channel attacks, or direct console access) than P assumes R to see, P's trust in the confidentiality might not be justified.

Assumption 4 For agents $P, R, Q, \ldots \in \mathbb{P}$ we assume that the following holds: $\forall x \in W_{RP} \exists y \in W_R : \mu(x) = \mu(y) \wedge (\lambda_{RP})^{-1}(\lambda_{RP}(x)) = (\lambda_R)^{-1}(\lambda_R(y))$. We assume the analogous assumptions to hold for arbitrary depths of nested trusted systems, e.g., that $\forall x \in W_{RQ_P} \exists y \in W_{RP} : \mu(x) = \mu(y) \wedge (\lambda_{RQ_P})^{-1}(\lambda_{RQ_P}(x)) = (\lambda_{RP})^{-1}(\lambda_{RP}(y))$, etc.

The above explanations refer to relations between initial knowledges and local views of the actual system S and those of S_P, P's conception of S. Analogous relations hold between systems of any nesting depth of trusted systems, i.e., for relations between S_P and S_{Q_P}, S_{Q_P} and $S_{Q'_{Q_P}}$, etc.

Section 7.2 discusses implications of these assumptions holding or not holding in real systems. The following theorems relate trust in confidentiality and satisfaction of confidentiality properties.

6.2 Implications between Trusted Properties

It is easy to show that Assumption 1 implies the following Lemma:

Lemma 1 Let S be a system as defined in Definition 7 with behaviour B, $P \in \mathbb{P}$ an agent, λ_P this agent's local view, and $W_P \supseteq B$ this agent's initial knowledge. Then for all $\omega \in B$ holds $\omega \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$.

The next theorem explains that authenticity is a stronger property than precedence. This is due to the fact that *precede* is defined on the system behaviour B, while *auth* is defined taking into account an agent's local view and initial knowledge of a system which can result in a bigger set of sequences of actions.

Theorem 1 For a system S as defined in Definition 7, actions $a, b \in \Sigma$, and agent $P \in \mathbb{P}$, $auth(a, b, P)$ holding in S implies that $precede(a, b)$ holds in S.

Proof 1 Let S be a system as defined in Definition 7 with behaviour B, $a, b \in \Sigma$, $P \in \mathbb{P}$, and let $auth(a, b, P)$ hold in S. Let us assume that $precede(a, b)$ does not hold in S. Then there is $\omega \in B$ with $b \in alph(\omega)$ and $a \notin alph(\omega)$. $b \in alph(\omega)$ and $auth(a, b, P)$ holding in S imply that $a \in alph(x)$ for all $x \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$. Since by Lemma 1 ω is one of the elements of $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$, it immediately follows that $a \in alph(\omega)$, which is a contradiction to the assumption. Hence $precede(a, b)$ holds in S.

Corollary 1 For a system S, actions $a, b \in \Sigma$ and agents $P, Q \in \mathbb{P}$, $trust(P, auth(a, b, Q))$ holding in S implies that $trust(P, precede(a, b))$ holds in S.

Proof 2 The assertion follows immediately from Theorem 1. Since $auth(a, b, P)$ implies $precede(a, b)$ in all systems, this implication holds in particular in the system S_P.

The next theorem shows that the authenticity of an action for an agent can be extended to a preceding action if the agent trusts in the precedence.

Theorem 2 For a system S as defined in Definition 7, actions $a, b, c \in \Sigma$ and an agent $P \in \mathbb{P}$, $auth(b, c, P)$ and $trust(P, precede(a, b))$ holding in S implies that

$auth(a, c, P)$ holds in S.

Proof 3 Let S be a system as defined in Definition 7, $a, b, c \in \Sigma$, $P \in \mathbb{P}$, and let $auth(b, c, P)$ hold in S. Then for all $\omega \in B$, if $c \in alph(\omega)$ then $b \in alph(x)$ for all $x \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$. Further, $trust(P, precede(a, b))$ holding in S means that for all $y \in W_P$, if $b \in alph(y)$ then $a \in alph(y)$. As $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P \subseteq W_P$, $a \in alph(x)$ in particular for all $x \in \lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$. Hence $auth(a, c, P)$ holds in S.

Corollary 2 For a system S, actions $a, b, c \in \Sigma$, and agents $P, Q \in \mathbb{P}$, $trust(Q, auth(b, c, P)) \wedge trust(Q, trust(P, precede(a, b)))$ holding in S implies that $trust(Q, auth(a, c, P))$ holds in S.

Proof 4 Analogously to the proof of Corollary 1, we use the fact that Theorem 2 applies to all systems, hence in particular to S_Q.

Lemma 2 For a system S as defined in Definition 7 and actions $a, b, c \in \Sigma$, $precede(a, b)$ and $precede(b, c)$ implies that $precede(a, c)$ holds in S.

Proof 5 Let S be a system as defined in Definition 7, $a, b, c \in \Sigma$. $precede(b, c)$ holding in S means that for all $\omega \in B$, if $c \in alph(\omega)$ then $b \in alph(\omega)$. Further, $precede(a, b)$ holding in S means that for all $\omega \in B$, if $b \in alph(\omega)$ then $a \in alph(\omega)$. This concludes that if $c \in alph(\omega)$ then $a \in alph(\omega)$ for all $\omega \in B$.

Theorem 3 For a system S and an agent $P \in \mathbb{P}$, $trust(P, precede(a, b) \wedge precede(b, c))$ holding in S implies that $trust(P, precede(a, c))$ holds in S.

Proof 6 Analogously to the proof of Corollary 1, we apply Lemma 2 to the system S_P.

Theorem 4 Let S be a system that satisfies Assumption 1, agent $P \in \mathbb{P}$ and actions $a, b \in \Sigma$. Then $trust(P, precede(a, b))$ implies $precede(a, b)$.

Proof 7 Let S be a system as defined in Definition 7, $a, b \in \Sigma$. $trust(P, precede(b, c))$ holding in S means that for all $\omega \in W_P$, if $b \in alph(\omega)$ then $a \in alph(\omega)$. Because according to Assumption 1 $B \subseteq W_P$, this is especially true for all $\omega \in B$ with $b \in alph(\omega)$.

Corollary 3 Let S be a system that satisfies Assumption 1, agents $P, Q \in \mathbb{P}$, and actions $a, b \in \Sigma$. Then $trust(P, trust(Q, precede(a, b)))$ implies $trust(P, precede(a, b))$.

Proof 8 Analogously to the proof of Theorem 4, we use Assumption 1 $W_P \subseteq W_{Q\,P}$ holding for the system S_P.

The following theorem covers the implications that exist between an agent's trust in confidentiality and the actual confidentiality of parameters.

Theorem 5 (Trust in Confidentiality) Let S be a system as defined in Definition 7 that satisfies Assumptions 1, 2, 3, 4, and $P \in \mathbb{P}$, $who \subseteq \mathbb{P}$. Then $trust(P, conf(\mathcal{A}(par), par, who))$ holding in S implies that $conf(\mathcal{A}(par), par, who)$ holds in S.

Proof 9 The property $trust(P, conf(\mathcal{A}(par), par, who))$ holding in S means that $conf(\mathcal{A}(par), par, who)$ holds in S_P, i.e., that for all agents $R \notin who$ exists $K \supseteq \mu(W_{RP})$ such that $\forall \omega \in W_P : \mu((\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) \cap W_{RP}) \supseteq p^{-1}(p(\mu((\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) \cap W_{RP}))) \cap K$. (Recall that μ corresponds to $\mathcal{A}(par)$, i.e., maps all actions not included in this set onto ε and further extracts the parameter par.)
In order for $conf(\mathcal{A}(par), par, who)$ to hold in S, it must be shown that for all agents $R \notin who$ exists an (L, M)–complete language $K' \subseteq (\Sigma_t \times M)^*$ with $K' \supseteq \mu(W_R)$ such that $\forall \omega \in B$ the following holds: $\mu((\lambda_R)^{-1}(\lambda_R(\omega)) \cap W_R) \supseteq p^{-1}(p(\mu((\lambda_R)^{-1}(\lambda_R(\omega)) \cap W_R))) \cap K'$. We show this inequality for $K' = K$.
Because of $B \subseteq W_P$, the first inequality holding for all $\omega \in W_P$ holds in particular for all $\omega \in B$.
The following diagram shows that in order to show the assertion (tagged with "?") from the inequality that holds (tagged with "!"), it is sufficient to show the inequalities (i) and (ii).

$$\begin{array}{ccc} \mu((\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) \cap W_{RP}) & \supseteq^{!} & p^{-1}(p(\mu((\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) \cap W_{RP}))) \cap K \\ \cap\!\mid \quad \text{(i)} & & \mid\!\cup \quad \text{(ii)} \\ \mu((\lambda_R)^{-1}(\lambda_R(\omega)) \cap W_R) & \supseteq^{?} & p^{-1}(p(\mu((\lambda_R)^{-1}(\lambda_R(\omega)) \cap W_R))) \cap K \end{array}$$

(i) Let $\omega \in B$ and $a \in (\Sigma_t \times M)^*$ with $a \in \mu((\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) \cap W_{RP})$. Then there exists $b \in (\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) \cap W_{RP}$ with $\mu(b) = a$. Since in particular $b \in W_{RP}$, by Assumption 4 it follows that exists $c \in W_R$ with $(\lambda_{RP})^{-1}(\lambda_{RP}(b)) = (\lambda_R)^{-1}(\lambda_R(c))$ and $\mu(b) = \mu(c)$. Further, $b \in (\lambda_{RP})^{-1}(\lambda_{RP}(\omega))$ implies $\lambda_{RP}(b) = \lambda_{RP}(\omega)$, hence $\omega \in (\lambda_{RP})^{-1}(\lambda_{RP}(b))$. From these two observations it follows that $\omega \in (\lambda_R)^{-1}(\lambda_R(c))$, i.e., $\lambda_R(\omega) =$

$\lambda_R(c)$ which in turn implies $c \in (\lambda_R)^{-1}(\lambda_R(\omega))$, hence $c \in (\lambda_R)^{-1}(\lambda_R(\omega)) \cap W_R$. Since further $\mu(b) = \mu(c)$ and $a = \mu(b)$, it follows the assertion.

(ii) Since by Assumption 3 exists a homomorphism h_P^R such that $\lambda_{RP}(\omega) = h_P^R(\lambda_R(\omega))$, it follows that $(\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) = (\lambda_R)^{-1}(h_P^{R^{-1}}(h_P^R(\lambda_R(\omega)))) \supseteq (\lambda_R)^{-1}(\lambda_R(\omega))$. Hence $(\lambda_{RP})^{-1}(\lambda_{RP}(\omega)) \cap W_{RP} \supseteq (\lambda_R)^{-1}(\lambda_R(\omega)) \cap W_{RP}$. By Assumption 2, it follows the assertion.

The following corollary targets specifically confidentiality implications in trusted systems and can be proven similarly:

Corollary 4 Let S be a system as defined in Definition 7, and S_P, S_{Q_P}, $S_{Q'_{Q_P}}, \ldots$ the resulting chain of nested trusted systems, all satisfying Assumptions 1, 2, 3, and 4. Then for all systems the implication of Theorem 5 holds analogously, in particular $\mathit{trust}(P, \mathit{trust}(Q, \mathit{trust}(Q', \mathit{conf}(\mathcal{A}(\mathit{par}), \mathit{par}, \mathit{who}))))$ implies $\mathit{trust}(P, \mathit{trust}(Q, \mathit{conf}(\mathcal{A}(\mathit{par}), \mathit{par}, \mathit{who})))$.

In the next section this notion of trust and the theorems and corollaries are applied to the example from Section 4, and, by doing so, identify trust assumptions that need to hold in order for the system to provide certain security properties.

7. The Example Formally

The following paragraph provide a formal specification of the example system. It has five agents, so $\mathbb{P} = \{Sensor_1, Sensor_2, Displ, TTP, User\}$. The parameter *data* used in the formalization of actions can have different values, e.g., *warm* and *cold*. Further, $i = 1, 2$ and $j \in \mathbb{N}$. The set Σ of actions consists of the following actions:

$\mathit{sense}(\mathit{Sensor}_i, \mathit{data})$	One of the sensors Sensor_i senses data.
$\mathit{send}(\mathit{Sensor}_i, \mathit{data}, \mathit{privK}_i, \mathit{sig}_j(\mathit{data}))$	Sensor_i signs data, using privK_i, and sends it.
$\mathit{recv}(\mathit{Displ}, \mathit{data}, \mathit{pubK}_i, \mathit{sig}_j(\mathit{data}))$	Displ receives a signature on data being validated with pubK_i.
$\mathit{show}(\mathit{Displ}, \mathit{data}, \mathit{Sensor}_i)$	Displ shows data having presumably been sent by Sensor_i to user.

This is a very abstract formalization of the system's actions chosen simply to facilitate understanding. The formalism works equally well with other notational conventions, e.g., $(\mathit{actionname}, \mathit{par}_1, \ldots, \mathit{par}_k)$.

In order to be able to reason about security properties of the formal model on the basis of the theorems introduced in the previous section, the model needs to satisfy Assumptions 1 to 4. While Section 6 briefly explains why these assumptions are reasonable, Section 7.2 investigates them in more detail in the context of the presented example.

An appropriate assumption regarding the user's local view is to assume that it keeps the action $\mathit{show}(\mathit{Displ}, \mathit{data}, \mathit{Sensor}_i)$ and maps all other actions onto the empty word, that is, the user can only see the data that is displayed to him/her, but cannot see the actual actions performed by other devices. As to the other agents, one can simply assume that they are able to see only their own actions. If we wanted to focus on the communication means (e.g., the devices being connected via a network bus) we could define their local views to keep the send and receive actions including the messages but removing the sender. **Figure 3** illustrates the requirement for the example system.

Using Definition 2, this requirement informally derived in Section 4 can be formally stated as follows:

$$\mathit{auth}(\mathit{sense}(\mathit{Sensor}_i, \mathit{data}), \mathit{show}(\mathit{Displ}, \mathit{data}, \mathit{Sensor}_i), \mathit{User}) \qquad \text{(P0)}$$

Note that we do not discuss here the quality of the data sensed by the sensor, i.e., the question of how near it represents reality (although SeMF allows to model this as well).

As explained in Section 4, a digital signature scheme can only establish a re-

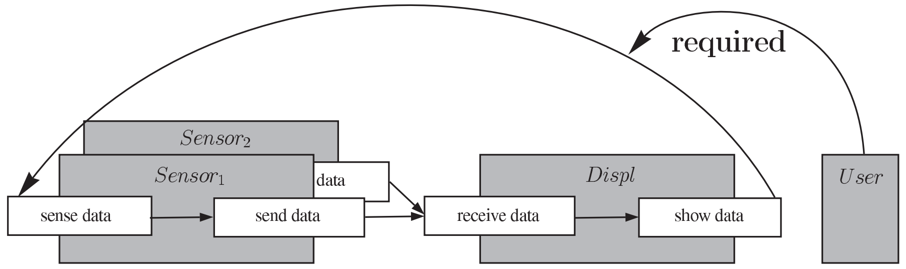

Fig. 3 Security requirement.

lation between signing/sending and receiving/verifying the data with respect to the key pair used in this process, but does not say anything about the agents in this respect. Hence one can formalize the property of an asymmetric digital signature scheme such that whenever a signature is positively verified with a certain public key, the corresponding private key must have been used to generate this signature:

$$\begin{aligned} &precede(\{send(P, data, privK_i, sig_j(data)) \mid P \in \mathbb{P}\}, \\ &\qquad \{recv(Q, data, pubK_i, sig_j(data)) \mid Q \in \mathbb{P}\}) \end{aligned} \tag{1}$$

Note that this property formalizes one interpretation of digital signature schemes like RSA that does not consider things such as weak keys, bad RNGs, etc. We will assume this property holds in every agent's trusted system, assuming that each of the agents trusts the mathematical properties of the digital signature scheme, namely this is:

$$\begin{aligned} &trust(User, precede(\{send(P, data, privK_i, sig_j(data)) \mid P \in \mathbb{P}\}, \\ &\qquad \{recv(Q, data, pubK_i, sig_j(data)) \mid Q \in \mathbb{P}\})) \end{aligned} \tag{2}$$

$$\begin{aligned} &trust(User, trust(Display, \\ &precede(\{send(P, data, privK_i, sig_j(data)) \mid P \in \mathbb{P}\}, \\ &\qquad \{recv(Q, data, pubK_i, sig_j(data)) \mid Q \in \mathbb{P}\}))) \end{aligned} \tag{3}$$

and so on.

However the requirement that shall be satisfied by using a digital signature scheme comprises more than just the knowledge that the signature has been generated before and with a specific key. It must be assured that a certain agent, e.g., $Sensor_i$, has signed and sent the signature generated with $privK_i$ (and not $Sensor_j$) .

In order for the signature scheme to actually provide this property, the system must satisfy several requirements: First, the sensor's private key needs to be confidential for the sensor, i.e., must not be known by any other agent. Second, the signed message must contain enough information to identify it as belonging to this particular sending action. In other words, it must not be possible for another signature generated by the sensor for different purposes to be confused with this one. In the following, we concentrate on the requirement regarding the confidentiality of the signature key and assume the other requirement to be satisfied. Confidentiality of the private key can be formalized as follows:

$$conf(\mathcal{A}(privK_i), privK_i, \{Sensor_i\}) \tag{4}$$

Here $\mathcal{A}(privK_i)$ denotes the set of actions that use the private key $privK_i$ which in our simple example consists of $send(Sensor_i, data, privK_k, sig_j(data))$ $(i, k = 1, 2,\ j \in \mathbb{N})$.

However, if none of the agents in the system knew about the confidentiality of $privK_i$ it would be of no use when reasoning about the authenticity of a message. Rather the agents must be aware of this fact. The example assumes the TTP to represent the roles of the manufacturer and certification authority of the sensor. Therefore by the time of deployment of a sensor (or already during production) the TTP assures itself of the *confidentiality of the $privK_i$ to the respective sensor*, formalized as:

$$trust(TTP, conf(\mathcal{A}(privK_i), privK_i, \{Sensor_i\})) \tag{5}$$

As stated before, in order for the display node to extend the authenticity of the sending action to the actual measuring action of the sensor, the display must trust that the sensor works correctly and only sends data that it has measured. As explained in Section 1, this type of trust in the correct functioning of a device can e.g., be achieved by trusted computing functionality. Similar to e.g., a Trusted Platform Module (TPM) the manufacturer or certificator TTP of a sensor device has the possibility to ensure that this device behaves in a certain manner. In the case of our sensor, this means that $Sensor_i$ will sign only those data with $privK_i$ and send it that it measured before. Using this approach, the resulting property can be formally stated as follows:

$$\begin{aligned} &trust(TTP, precede(sense(Sensor_i, data), \\ &\quad send(Sensor_i, data, privK_i, sig_j(data)))) \end{aligned} \tag{6}$$

As it was stated above, the TTP is assumed to issue certificates to be incorporated at the display, describing the sensor. These certificates will basically represent the two above mentioned trust properties 5 and 6 regarding TTP's trust and transfer them as trust assertions to the display that will adopt them. As a result, the display itself will trust in these TTP trust properties, formalized as:

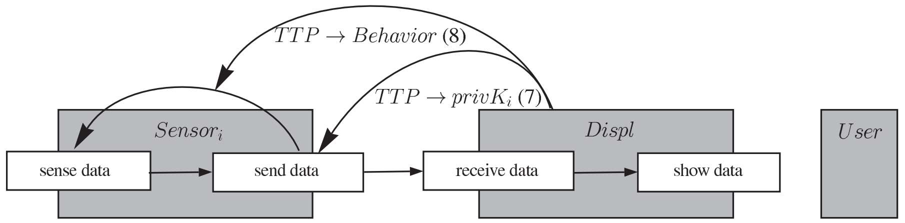

Fig. 4 Security properties regarding the Display.

$$trust(Displ, trust(TTP, conf(\mathcal{A}(privK_i), privK_i, \{Sensor_i\}))) \tag{7}$$

$$\begin{aligned} &trust(Displ, trust(TTP, precede(sense(Sensor_i, data), \\ &\quad send(Sensor_i, data, privK_i, sig_j(data))))) \end{aligned} \tag{8}$$

The resulting security properties are illustrated in **Fig. 4**.

However, as already explained, the stakeholder that requires the information to be authentic is the user rather than the display. Thus the user has to trust in the correct functioning of the display node. This means on the one hand that the user trusts the display only to show data it has received. This can be captured with the following formalization:

$$\begin{aligned} &trust(User, precede(recv(Displ, data, pubK_i, sig_j(data)), \\ &\quad show(Displ, data, Sensor_i))) \end{aligned} \tag{9}$$

On the other hand, the user's trust into the correct functioning of the display includes the user's trust that the display node establishes its own trust into the identification of the signer and the correct functioning of this signer. This can be formalized as follows:

$$trust(User, trust(Displ, trust(TTP, conf(\mathcal{A}(privK_i), privK_i, \{Sensor_i\})))) \tag{10}$$

$$\begin{aligned} &trust(User, trust(Displ, trust(TTP, precede(sense(Sensor_i, data)), \\ &\quad send(Sensor_i, data, privK_i, sig_j(data))))) \end{aligned} \tag{11}$$

Further possible mechanisms that can ensure that these three properties hold are not discussed here. These may include a good reputation of the manufacturer or the display being a "quality product", means of contractual binding, etc. However, in a concrete security engineering process all properties that are assumed to hold must be substantiated and evaluated e.g., by risk analysis techniques.

Finally, as explained at the beginning of this section, the user's local view keeps the action $show(Displ, data, Sensor_i)$, i.e., the data being shown on the display is visible to the user. We further assume that measures are in place that provide trustworthiness of messages shown by the display for the user, i.e., the user can trust that whenever he/she is shown a message, it originates from the display. This results in the following property:

$$auth(show(Displ, data, Sensor_i), show(Displ, data, Sensor_i), User) \tag{12}$$

7.1 Reasoning with Trust

The previous section discussed the appropriateness of some basic assumptions regarding the example system and introduced some security properties of a signature mechanism deduced from these assumptions. In this section we will use the theorems introduced in Section 6.2 to prove that these properties and assumptions result in satisfaction of property P0 required to hold for the example system.

We start from Property 10 which states that the user trusts the display's trust in the TTP's trust regarding the confidentiality of $privK_i$ to $Sensor_i$:

$$\begin{aligned} &trust(User, trust(Displ, trust(TTP, \\ &conf(\mathcal{A}(privK_i), privK_i, \{Sensor_i\})))) \end{aligned} \tag{13}$$

As Assumptions 1 to 4 may be appropriately assumed to hold, Corollary 4 can be applied to conclude that the user trusts the display's direct trust in the confidentiality of $privK_i$:

$$trust(User, trust(Displ, conf(\mathcal{A}(privK_i), privK_i, \{Sensor_i\}))) \tag{14}$$

As defined in Property 3, the display trusts the mathematical properties of the digital signature algorithm, namely that a signature that is verified with a specific key must have been signed with the respective private key. One may now conclude that in combination with Property 14 the display trusts that it was $Sensor_i$ that performed the signing action whenever the display verifies a signature with $pubK_i$:

Lemma 3 If Properties 1 and 14 hold, then the following property holds:

$$\begin{aligned} &trust(User, trust(Displ, precede(send(Sensor_i, data, privK_i, sig_j(data)),\\ &\qquad recv(Displ, data, pubK_i, sig_j(data))))) \end{aligned} \tag{15}$$

Proof 10 Because Property 3 holds, the display trusts that whenever it verifies a signature with some public key there must have been some agent who generated the signature:

$$\begin{aligned} &\exists P \in \mathbb{P} : trust(User, trust(Displ, precede(send(P, data, privK_i, sig_j(data)),\\ &\qquad recv(Displ, data, pubK_i, sig_j(data))))) \end{aligned}$$

However, for all $P \in \mathbb{P} \setminus \{Sensor_i\}$ it would be a contradiction to Property 14 to perform the action $send(P, data, privK_i, sig_j(data))$ that directly involves the usage of the private key $privK_i$. Hence the Proposition holds.

The user trusts that the display's actions are authentic to the display:

$$\begin{aligned} &trust(User, auth(recv(Displ, data, pubK_i, sig_j(data)),\\ &\qquad recv(Displ, data, pubK_i, sig_j(data)), Displ)) \end{aligned} \tag{16}$$

With this trivial property, Corollary 2 can be applied which can be interpreted as:

The display trusting into the confidentiality of the sensor's signature key implies the authenticity for the display that a sign and send action is performed by $Sensor_i$ whenever it has received and verified $Sensor_i$'s signature. Since the user trusts the display regarding its trust into the confidentiality of the sensor's signature key, the user also trusts authenticity of the sensor's signature to the display. We can conclude that the following property is met by the system:

$$\begin{aligned} &trust(User, auth(send(Sensor_i, data, privK_i, sig_j(data)),\\ &\qquad recv(Displ, data, pubK_i, sig_j(data)), Displ)) \end{aligned} \tag{R2}$$

Focusing now on Property 11 that expressed the trust of the user in the display's trust in the trust of the *TTP* in the behavioral property of the sensor, we may apply Corollary 3 that allows us to conclude the trust of the user in the display's trust in the sensor's internal behavior:

$$\begin{aligned} &trust(User, trust(Displ, precede(sense(Sensor_i, data),\\ &\qquad send(Sensor_i, data, privK_i, sig_j(data))))) \end{aligned} \tag{17}$$

This behavioral property can now be combined with Property R2 that refers to the authenticity of the message transfer from the sensor to the display trusted by the user. The combination utilizing Corollary 2 can therefore be used to conclude that the user trusts in the authenticity of the complete functional chain from the sensing of data by the sensor to the receiving of this data over the network by the display:

$$\begin{aligned} &trust(User, auth(sense(Sensor_i, data),\\ &\qquad recv(Displ, data, pubK_i, sig_j(data)),\\ &\qquad Displ\ )) \end{aligned} \tag{18}$$

Because of the user's trust into the authenticity of this functional chain, Corollary 1 can be used to conclude that the user trusts in the precedence of the respective sense and receive actions.

$$\begin{aligned} &trust(User, precede(sense(Sensor_i, data),\\ &\qquad recv(Displ, data, pubK_i, sig_j(data)))) \end{aligned} \tag{19}$$

This intermediate step together with Property 9 is illustrated in **Fig. 5**. Theorem 3 allows to combine these to a precedence property trusted by the user that spans over the whole functional chain from sensing to displaying of the data:

$$\begin{aligned} &trust(User, precede(sense(Sensor_i, data),\\ &\qquad show(Displ, data, Sensor_i))) \end{aligned} \tag{20}$$

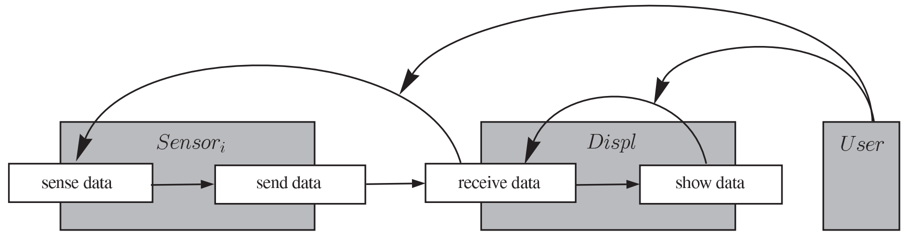

Fig. 5 Intermediate proof step.

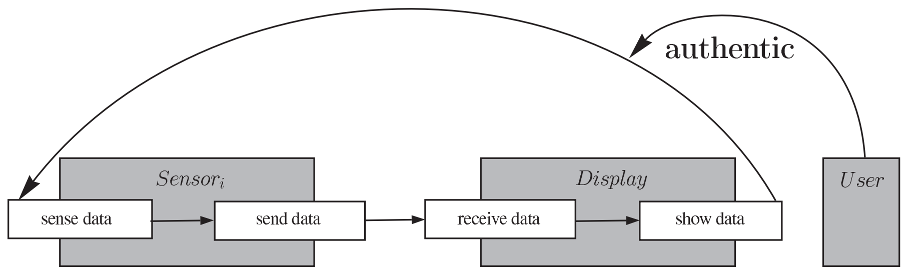

Fig. 6 Result.

Finally the application of Theorem 2 to Properties 12 and 20 allows to deduce the security property the example system shall provide, namely that the data shown to the user on the display is authentically the data that was measured by the sensor:

$$\begin{aligned} &auth(sense(Sensor_i, data),\\ &show(Displ, data, Sensor_i), User\ ) \end{aligned} \tag{21}$$

This resulting security property is illustrated in **Fig. 6**.

This proves that the user can be assured that data displayed to him/her is the same that was measured by the sensor before. The proof is based on two important aspects: First, the properties the system is assumed to provide, substantiated by security mechanisms assumed to be used by the system, and second, the relations between these properties that are proven to hold in general in the previous section. This proof constitutes only one way to achieve this result. In fact, different theorems can be derived using the same assumptions and leading to different proof paths. Further, since the assumptions constitute sufficient conditions, systems that do not satisfy them might still provide the same authenticity property.

7.2 Analysis of Assumptions

The assumptions used throughout the proofs of the previous section reveal what is implicitly assumed to hold in systems that use digital signature schemes or certificate-based key distribution for trust assertions on the confidentiality of keys or behavior of agents. The elaboration of the formally expressed assumptions allows us to pinpoint critical parts of the system and to explicitly analyse the plausibility of these assumptions. The following paragraphs discuss implications for a system in which these assumptions are inadequate.

Assumption 1 states that (i) an agent will not assume wrong conditions about the world, and (ii) that an agent cannot assume another agent to know more than the agent itself does. In the example, the TTP assumes that the private key $privK_i$ stays confidential to the sensor it belongs to, and that this knowledge is transferred through a certificate to the user-trusted display. If there was however an attack on the sensor that the TTP knows about, but the user does not, then the user would of course not be able to extend his/her conception of the TTP's knowledge with this information unless he/she learned about this information himself in the first place. This is the reason why Certificate Revocation Lists are essential in any PKI: They enable users to extend their own knowledge about security breaches in the system, since it is not possible to “magically” incorporate the full knowledge of a different agent into another subject's conception.

Assumptions 2 and 3 state that an agent must not grant another agent more knowledge or a more precise local view than this other agent really has. The user could for example falsely assume the display to have received certificates from the TTP informing it about the trustworthy behavior of the sensor, whilst in reality this is not the case. The user would follow the argumentation laid out in the previous section and believe the data that is shown by the display to originate from the sensing action, whilst in reality the sensor has sensed different data then the one it sent. This is a typical problem in PKIs when it comes to the interpretation of a certificate. The Trusted Computing Group for example made very clear distinctions between the meaning of an EK certificate and a platform certificate though they need to be used at the same time in order to derive a meaningful assertion about the system's runtime behavior [22]. The same interpretation problem arises when the user assumes the display to be able to interpret (i.e., see in its local view) the content of a certificate, i.e., the data being signed. If this was not the case, whilst the user believes it, the user would falsely assume the authenticity of the displayed data.

Assumption 4 targets the direct opposite of the assumptions above. It basically states that the knowledge and local view of agents must not be underestimated.

This assumption targets directly the nature of our concept of parameter confidentiality. In our example, this applies to the confidentiality of $privK_i$. If the TTP falsely assumed that the private key cannot be observed by a potential attacker, this would of course lead to a security breach. These kinds of attacks that include side-channel attacks as well as offline attacks against key storages are a direct contradiction to the assumption's part concerning local views. At the same time, the attacker's knowledge must not be underestimated. If the attacker was e.g., aware of a weakness in the random number generator used during the creation of $privK_i$ (such as the Debian RNG-Bug), then this would lead to a security breach as well. However, violation of Assumption 4 has no undesired effect on authenticity. If the attacker's knowledge about the trustworthiness of the display was underestimated for example, authenticity would still hold.

8. Conclusions and Future Work

Satisfaction of security properties always depends on the overall properties, assumptions and trust relations of a system. All of these have to be considered in security assessment as well as in system engineering processes. The formal notion of trust introduced in Ref. 2) and used in this paper to express trust requirements concerning authenticity and confidentiality can serve as a basis to exactly express assumptions and trust relations within a formal framework, namely the Security Modeling Framework SeMF. The example from sensor networks with a focus on trust in cryptographic keys and public key infrastructures demonstrates formal reasoning on trust and shows the formal derivation of explicitly specified trust assumptions that are usually implicitly assumed to hold. PKIs for example use implicit assumptions regarding e.g., trust of a TTP in the confidentiality of an agent's private key. Applying the approach to these trust requirements finally extracts those trust assumptions that cannot be further substantiated within a formal model of the technical system, such as those resulting from identification mechanisms used by the TTP to identify the owner of a private key.

Future work includes extending the approach to further types of security properties formalized within SeMF and also applying the approach to real-world examples to explore the applicability and to find proper interfaces to semi-formal or informal risk analysis methods and tools.

Acknowledgments Part of this work was accomplished within the project EVITA 224275 funded by the European Commission.

References

1) Gürgens, S., Ochsenschläger, P. and Rudolph, C.: On a formal framework for security properties, *International Computer Standards & Interface Journal (CSI), Special Issue on Formal Methods, Techniques and Tools for Secure and Reliable Applications*, Vol.27, No.5, pp.457–466 (2005).
2) Fuchs, A., Gürgens, S. and Rudolph, C.: A Formal Notion of Trust–Enabling Reasoning about Security Properties, *Proc. 4th IFIP WG 11.1 International Conference on Trust Management* (2010).
3) Trusted Computing Group: TCG TPM Specification 1.2 Revision 103, http://www.trustedcomputinggroup.org (2006).
4) Muskens, J., Alonso, R., Yhang, Z., Egelink, K., Larranaga, A. and Gouder, A.: Trust4All Trust Framework and Mechanisms, *ITEA Trust4All* (2005).
5) Jøsang, A., Ismail, R. and Boyd, C.: A survey of trust and reputation systems for online service provision, *Decision Support Systems*, Vol.43, No.2, pp.618–644 (2007).
6) McKnight, D. and Chervany, N.: The Meanings of Trust, Technical Report, University of Minnesota, Management Information Systems Reseach Center (1996).
7) Grandison, T. and Sloman, M.: A Survey of Trust in Internet Applications, *IEEE Communications Surveys and Tutorials*, Vol.3, No.4, pp.2–16 (2000).
8) Carbone, M., Nielsen, M. and Sassone, V.: A Formal Model for Trust in Dynamic Networks, pp.54–61 (2003). RS-03-4.
9) Demolombe, R.: Reasoning about trust: A formal logical framework, *Lecture Notes in Computer Science*, pp.291–303 (2004).
10) Burrows, M., Abadi, M. and Needham, R.: A Logic of Authentication, *ACM Trans. Comput. Syst.*, Vol.8 (1990).
11) Syverson, P. and van Oorschot, P.: On unifying some cryptographic protocol logics, *IEEE Symposium on Security and Privacy*, pp.14–28 (1994).
12) Abadi, M. and Tuttle, M.: A Semantics for a Logic of Authentication, *10th Annual ACM Symposium on Principles of Distributed Computing*, Montreal, Canada, pp.201–216 (1991).
13) Gong, L., Needham, R. and Yahalom, R.: Reasoning about Belief in Cryptographic Protocols, *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, pp.234–248, IEEE Press (1990).
14) Gürgens, S.: SG Logic–A formal analysis technique for authentication protocols, *Security Protocols*, *LNCS*, Vol.1316, pp.159–176, Springer Verlag (1997).
15) Ellison, C. and Schneier, B.: Ten risks of PKI: What you're not being told about public key infrastructure, *Computer*, Vol.16, No.1, p.1 (2000).

16) Ozols, M., Cant, T., Liu, C. and Henderson, M.: Formalization of public key infrastructures (2001).
17) Huang, J. and Nicol, D.: A calculus of trust and its application to PKI and identity management, *Proc. 8th Symposium on Identity and Trust on the Internet*, pp.23–37, ACM (2009).
18) Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J.: Tropos: An agent-oriented software development methodology, *Autonomous Agents and Multi-Agent Systems*, Vol.8, No.3, pp.203–236 (2004).
19) Eilenberg, S.: *Automata, Languages and Machines*, Academic Press, New York (1974).
20) Gürgens, S., Ochsenschläger, P. and Rudolph, C.: Parameter confidentiality, *Informatik 2003–Teiltagung Sicherheit*, Gesellschaft für Informatik (2003).
21) Gürgens, S., Ochsenschläger, P. and Rudolph, C.: Abstractions preserving parameter confidentiality, *European Symposium on Research in Computer Security* (*ESORICS 2005*), pp.418–437 (2005).
22) Trusted Computing Group: TCG Credential Profiles, Specification Version 1.1, Technical Report (2007).

(Received October 31, 2010)
(Accepted January 14, 2011)
(Original version of this article can be found in the Journal of Information Processing Vol.19, pp.274–291.)

Andreas Fuchs studied computer science at the Technical University of Darmstadt and the University of Massachusetts (Amherst, USA) and received his Diploma at the former. Since then, he is working as a scientist at the Fraunhofer Institute for Secure Information Technology. His research concentrates on formal methods, information security, trusted computing and security engineering, includeing Remote Attestation Techniques and pattern based security application. His current work includes topics from security engineering for embedded systems as well as service oriented architectures.

Sigrid Gürgens received her Ph.D. in Mathematics at the Technische Hochschule Darmstadt in 1992. Since 1988 she is working as a scientist at the Fraunhofer Institute for Secure Information Technology. Her research focuses on formal methods, information security and cryptography. In particular she worked on tool-supported analysis and design of security protocols, and was engaged in the security validation of scenarios based on the V1.2 specification of the Trusted Platform Module (TPM) and has contributed to the work of the TPM working group. Currently she is working on the development of formal security models, on methods for the specification and refinement of security properties, on methods for proving such properties.

Carsten Rudolph received his Ph.D. in Computer Science at Queensland University of Technology, Brisbane in 2001. Since then, he is working at the Fraunhofer SIT where he is now head of the research department Security Modeling and Model Validation. His research concentrates on information security, formal methods and cryptographic protocols. Among other activities he recently worked on the development of a library of security solutions for AmI environments in the IST project SERENITY and he currently contributes as an invited expert to the standardisation of the trusted platform module TPM in the Trusted Computing Group.