A Formal Notion of Trust – Enabling Reasoning about Security Properties

Andreas Fuchs and Sigrid Gürgens and Carsten Rudolph *

Fraunhofer Institute for Secure Information Technology SIT,
Rheinstrasse 75, 64295 Darmstadt, Germany
{andreas.fuchs,sigrid.guergens,carsten.rudolph}@sit.fraunhofer.de

Abstract. 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 for the more general notion of trust needed when reasoning about security in ICT systems. In this paper we present a formal definition of trust to be able to exactly express trust requirements from the view of different entities involved in the system and to support formal reasoning such that security requirements, security and trust mechanisms and underlying trust assumptions can be formally linked and made explicit. Integrated in our Security Modeling Framework this formal definition of trust can support security engineering processes and formal validation and verification by enabling reasoning about security properties w.r.t. 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 it 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 to understand trust establishment and degrees of trustworthiness in these application domains, they cannot be used for

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

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. 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 have trust in contradictory properties. 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 certification authorities).

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 a specific authenticity property holds in a system. This example is used to explain our new notion of trust and to show how reasoning can lead to refined trust properties that express underlying trust assumptions and assumptions on security mechanisms that are not further refined within the model.

The following two sections first provide some terminology and then briefly discuss its relation to existing notions of trust. In Section 4 we introduce an example that will be used throughout the rest of the paper and that imposes some interesting questions related to trust. Section 5 then gives a brief summary of our Security Modeling Framework SeMF. Based on this, we present our formal notion of trust in Section 6 and use it to formally prove some security properties of our example in Section 7. Finally we present our conclusions in Section 8.

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) [2]. Therefore, we do not intend to contribute to the philosophical discussions on trust or the relation between trust and reputation.

Within our formal framework 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 which 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 the 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 [3]:

> "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. [4] 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 [5])" 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. [4]. They explain these concepts further by means of temporary reputation based systems such as eBay. Another survey that focusses particularly on trust management systems is given by Grandison and Sloman [6].

A second branch of research focusses on formal models that capture certain aspects of trust. In [7] Carbone et al. introduce a formal model of trust that focusses 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. Delombe [8] 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 aim 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 [9] 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 [10,11,12,13]). 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 is motivated by the work in the area of trusted computing [2] where 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 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.

4 An Example

In this section we introduce a very simple use case that includes a security requirement 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 our 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.

However, we discuss the security properties and resulting trust requirements only informally in this section in order to give an understanding of the practical implication of our notion of trust. In the subsequent sections we will provide a brief introduction to our formal Security Modeling Framework (SeMF), will introduce the formal definition of trust, some resulting theorems, and will revisit the example formally.

In order to explain our approach we will use the scenario illustrated in Figure 1.

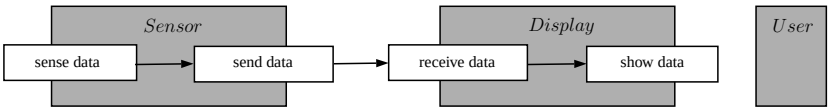

Fig. 1. Example System

Our example system consists of two active nodes and an end user. *Sensor* is a node deployed somewhere in the system that performs measurements (e.g. measures the temperature outside a house) and sends the resulting data over the network. *Display* is the second 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 being shown some data, wants this data to be indeed measured by the 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 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, such as digital signature schemes or message authentication codes (MACs). However, whatever mechanism is used, the user, a human being, cannot validate a digital signature or MAC, this has to be done by the Display node. Therefore it is the Display node that can be assured of the authenticity and not the user. Further 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, it can be sure that the signature or MAC was generated by the Sensor. Yet, this assurance does not extend 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).

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 property provided by a digital signature or MAC can then be expressed as follows:

P1 *It is authentic for the Display node that the data received is the data that was sent by the Sensor.*

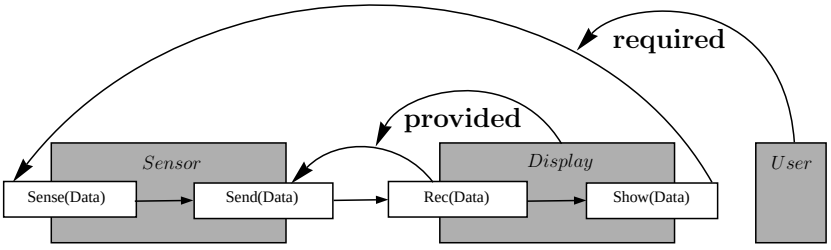

Fig. 2. Discrepancy between required and provided property

The discrepancy between property P0 we want the system to provide and property P1 that is actually provided by a digital signature or MAC is illustrated in Figure 2. For achieving P0 we need the system to provide more properties: In order for the Display to "extend" the authenticity of the send action by the

Sensor node to the actual measuring action, the Display must trust the Sensor that it only sends data it has measured. Further, the user must trust the Display to show only what it has received and verified. Hence we need a concept of trust that allows to relate agents to the system properties they trust in and that enables formal reasoning about these properties.

> 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 of those parts of the Security Modelling Framework SeMF that are the basis for the notion of trust presented in this paper. We then explain the concept of a property being fulfilled in an agent's conception of a system, the basis for our definition 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 [14].

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 could not 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 that an action $send(sender, message)$ occurred but cannot see the message, in which case $\lambda_P(send(sender, message))$ $= send(sender)$.

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 from P's local view after ω has happened. 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 [1] we have introduced a variety of definitions of security properties (e.g. authenticity, proof of authenticity, confidentiality). Our concept of trust introduced in the next section applies to all of them, however, we will use our notion of authenticity as a demonstrating example.

We call a particular action a authentic for an agent P if in all sequences that P considers to have possibly happened 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 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 actions $a, b \in \Sigma$, $precede(a, b)$ holds in S if for all $\omega \in B$ with $b \in alph(\omega)$ it follows that $a \in alph(\omega)$.*

6 A Formal Definition of Trust

In the previous section we have presented several factors that are important regarding security properties holding or not holding in a system. Accordingly we include these in the formal definition of a system as follows:

Definition 4 (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* $\mathbb{V} = \{\lambda_X : \Sigma^* \to (\Sigma_X)^* | X \in \mathbb{P}\}$ *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 defer 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_{XP}) and local views (λ_{XP}) may defer from the actual initial knowledges and local views of the system S. This motivates the following definition.

Definition 5 (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* $\mathbb{V}_P = \{\lambda_{XP} : \Sigma^* \to (\Sigma_{XP})^* | X \in \mathbb{P}\}$ *of agent* P*'s conception of agents' local views of* S*, and a set* $\mathbb{W}_P = \{W_{XP} \subseteq \Sigma^* | X \in \mathbb{P}\}$ *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 6 (Trusted Property) *Let* prop *be any property that refers to a system as defined in Definition 4. An agent* $P \in \mathbb{P}$ *trusts in* prop *to hold in a system* S*, denoted by* trust(P, prop)*, iff* 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 our 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.

6.1 Implications between Properties

In this section we present and prove 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.

We use the following Assumption 1 to model that agents do not falsely exclude behaviour that can actually happen in the system. The correctness of this assumption must be verified during the formalization of a real system as a SeMF system model, violations identify flaws of the real system's design. Assumption 1 further models the fact that an agent P cannot assign an agent Q more knowledge about a system's behaviour than P itself knows about it. Note that the more an agent knows the smaller its initial knowledge.

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

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

Lemma 1 *Let S be a system as defined in Definition 4 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 precede. 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 4, 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. *Let S be a system as defined in Definition 4 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 $a \in alph(\omega)$, 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. *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 4, 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. *Let S be a system as defined in Definition 4, $a, b, c \in \Sigma, P \in \mathbb{P}$, 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. *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 4 and actions $a, b, c \in \Sigma$, $precede(a, b)$ and $precede(b, c)$ implies that $precede(a, c)$ holds in S.*

Proof. *Let S be a system as defined in Definition 4, $a, b, c \in \Sigma$. $precede(b, c)$ 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. *Analogously to the proof of Corollary 1, we apply Lemma 2 to the system S_P.*

In the next section we will apply the notion of trust and the above theorems to the example introduced in 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

We now specify our example system formally. It has three agents, so $\mathbb{P} = \{Sensor, Display, User\}$. The set Σ of actions consists of *Sensor-sense*$(data)$, *Sensor-send*$(data)$, *Display-rec*$(data)$, *Display-show*$(data)$ and we assume that the parameter *data* can have different values, e.g. *warm* and *cold*. This is a very abstract formalization of the system's actions chosen simply to facilitate understanding. Our formalism works equally well with other notational conventions, e.g. $(actionname, par_1, \ldots, par_k)$. Our formal model of the system needs to satisfy Assumption 1, in particular W_{User} must be a subset of $W_{User\,Display}$. As for our abstract system we simply assume equality. Further, agents' initial knowledges and local views must comply with the mechanisms and resulting properties discussed below. However, for our purposes we do not need to specify concrete initial knowledges of agents, and regarding the agents' local views it is sufficient to specify that the user's local view keeps the action *Display-show*$(data)$ and

maps all other actions onto the empty word, that is, the user can only see the data that is displayed to him/her. As to the other agents we simply assume them to be able to see only their own actions.

Figure 3 illustrates the requirement we have for the example system.

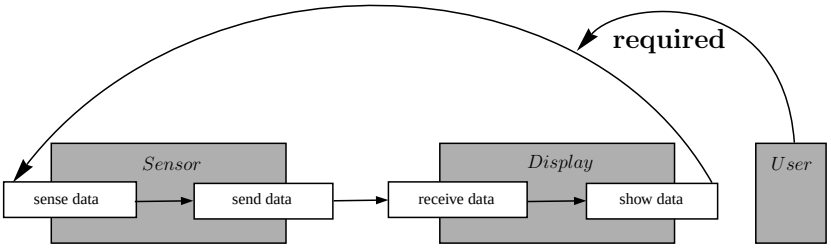

Fig. 3. Security Requirement

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

$$auth(Sensor\text{-}sense(data), Display\text{-}show(data), User) \tag{P0}$$

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 our framework allows to model this as well).

As explained in Section 4, a digital signature scheme (Sig) or message authentication code (MAC) can only establish a relation between signing/sending and receiving/verifying the data. Hence the property P1 informally stated in Section 4 can be formalized as follows:

$$auth(Sensor\text{-}send(data), Display\text{-}rec(data), Display) \tag{1}$$

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 the Sensor that it 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. This involves a Trusted Platform Module (TPM) to be integrated in the Sensor that measures and signs the configuration of the Sensor, using a signature key certified by a trusted authority. The details of the very complex behaviour of a TPM are not the topic of this paper, thus we refer the reader to [15]. Using trusted computing functionality, the resulting property can be formally stated as follows:

$$trust(Display, precede(Sensor\text{-}sense(data), Sensor\text{-}send(data))) \tag{2}$$

The resulting security properties are illustrated in Figure 4.

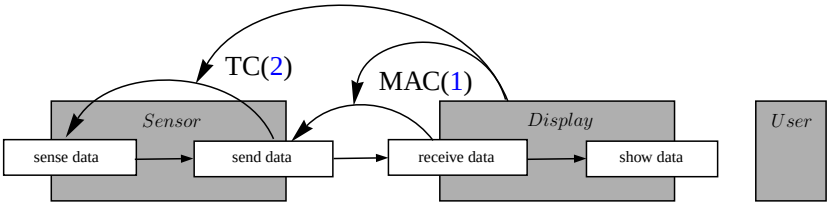

Fig. 4. Security Properties regarding the Display

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 has to trust in that the Display node establishes its own trust into the correct functioning of the Sensor and verifies the Sensor's signature when receiving the message. This can be formalized as follows:

$$trust\big(User, trust(Display, precede(Sensor\text{-}sense(data), Sensor\text{-}send(data))) \\ \wedge auth(Sensor\text{-}send(data), Display\text{-}rec(data), Display)\big) \tag{3}$$

On the other hand, the user's trust in the correct functioning of the Display includes trust in the Display only showing data it has received. This can be captured with the following formalization:

$$trust(User, precede(Display\text{-}rec(data), Display\text{-}show(data))) \tag{4}$$

We do not discuss here possible mechanisms that can ensure that properties 3 and 4 hold. 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. The above two properties are illustrated in Figure 5.

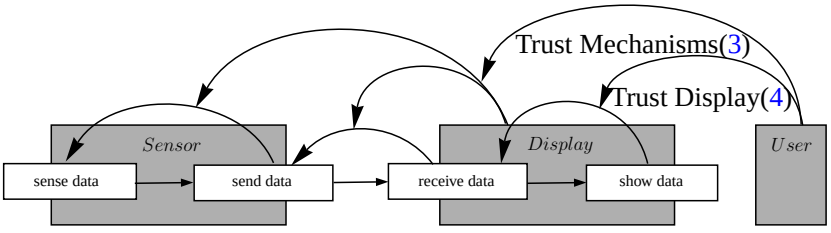

Fig. 5. Security Properties regarding the user

Finally, as explained at the beginning of this section, the User's local view keeps the action *Display-show*, i.e. the data being shown on the Display is visible to the user. This results in the following property:

$$auth(Display\text{-}show(data), Display\text{-}show(data), User) \tag{5}$$

7.1 Reasoning with Trust

In the previous section we have discussed various mechanisms and introduced the security properties provided by these mechanisms. In this section we will use the Theorems introduced in Section 6 to prove that these properties result in property P0 required to hold for our example system.

Starting from Property (3) that describes the user's trust into the trusted computing and MAC mechanisms, we can apply Corollary 2 and derive the following property:

$$trust(User, auth(Sensor\text{-}sense(data), Display\text{-}rec(data), Display)) \tag{6}$$

Applying Corollary 1 we can conclude:

$$trust(User, precede(Sensor\text{-}sense(data), Display\text{-}rec(data))) \tag{7}$$

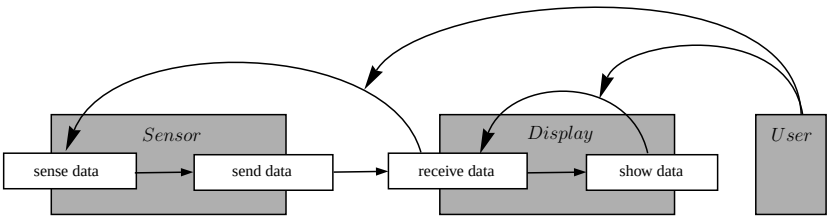

Fig. 6. Intermediate Proof Step

This intermediate step (7) together with Property 4 is illustrated in Figure 6. Theorem 3 allows to combine these to:

$$trust(User, precede(Sensor\text{-}sense(data), Display\text{-}show(data))) \tag{8}$$

Finally the application of Theorem 2 to Properties (5) and (8) implies

$$auth(Sensor\text{-}sense(data), Display\text{-}show(data), User) \tag{9}$$

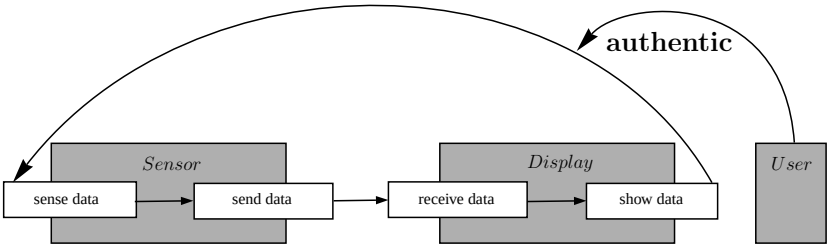

Fig. 7. Result

The resulting security property is illustrated in Figure 7.

This proves that the User can be assured that the data displayed is the same that was measured by the Sensor before. Our proof is based on two important aspects: the properties we assume the system to provide, substantiated by security mechanisms we assume the system to use, and the relations between these properties that we have proven to hold. This proof constitutes only one way to achieve this result, there are other ways that use different relations between properties which will be introduced in forthcoming papers.

8 Conclusions and Future Work

In this paper we have presented a formal definition of trust which reflects a simple understanding of the concept: An agent trusts in a specific property to hold in a system if it holds in its conception of the system. By integrating this formal trust concept into the SeMF Security Modeling Framework we have proven some relations between trust and specific authenticity properties. Using these we have exemplarily proven that an agent's wide-spanning authenticity property that can not be provided by current security mechanisms can be refined towards security properties that are provided by applicable security mechanisms. The refinement introduces other agents and makes use of trust relations between them. This process can be continued by substantiating certain trust assumptions through introducing further security mechanisms. Our approach has several advantages: It allows to formally prove certain security properties to hold in a system, and it extracts those trust assumptions that need to be substantiated by means beyond our formal framework (such as legal contracts). It further supports traceability: It allows to precisely identify the security properties that may be violated if a specific security mechanism is removed from the system (e.g. if the generation of a digital signature is removed because it turns out to be too slow) or if a trust assumption regarding an agent's behaviour is violated. The notion of trust is irrespective of underlying trust and security mechanisms. This makes it useful for risk analysis and management with respect to the comparison of different implementations.

We are still at the beginning of our work. Transitivity of trust for example can not be assumed to hold in general, sufficient conditions have to be found. More

theorems relating trust in e.g. confidentiality, authenticity, and non-repudiation will allow to capture a wider variety of security mechanisms substantiating trust. A further interesting topic is to find conditions under which abstractions preserve certain trust properties. The aim of our work is to enable a security engineering process that takes as input a high level security property and provides as result a set of conditions to implement this property and the identification of the assumptions that need to hold.

References

1. Gürgens, S., Ochsenschläger, P., 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 **27**(5) (June 2005) 457–466
2. Trusted Computing Group: TCG TPM Specification 1.2 revision 103. www.trustedcomputing.org (2006)
3. Muskens, J., Alonso, R., Yhang, Z., Egelink, K., Larranaga, A., Gouder, A.: Trust4All Trust Framework and Mechanisms. ITEA Trust4All (2005)
4. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems **43**(2) (2007) 618–644
5. McKnight, D., Chervany, N.: The Meanings of Trust. Technical report, University of Minnesota, Management Information Systems Reseach Center (1996)
6. Grandison, T., Sloman, M.: A survey of trust in internet applications. IEEE Communications Surveys and Tutorials **3**(4) (2000) 2–16
7. Carbone, M., Nielsen, M., Sassone, V.: A formal model for trust in dynamic networks. (September 2003) 54–61 RS-03-4.
8. Demolombe, R.: Reasoning about trust: A formal logical framework. Lecture notes in computer science (2004) 291–303
9. Burrows, M., Abadi, M., Needham, R.: A Logic of Authentication. ACM Transactions on Computer Systems **8** (1990)
10. Syverson, P., van Oorschot, P.: On unifying some cryptographic protocol logics. In: IEEE Symposium on Security and Privacy. (May 1994) 14–28
11. Abadi, M., Tuttle, M.: A Semantics for a Logic of Authentication. In: Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Canada. (August 1991) 201–216
12. Gong, L., Needham, R., Yahalom, R.: Reasoning about Belief in Cryptographic Protocols. In: Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, IEEE Press (1990) 234–248
13. Gürgens, S.: SG Logic – A formal analysis technique for authentication protocols. In: Security Protocols. Volume 1316 of LNCS., Springer Verlag (April 1997) 159 – 176
14. Eilenberg, S.: Automata, Languages and Machines. Academic Press, New York (1974)
15. Mitchell, C., et al.: Trusted Computing. Institution of Engineering and Technology (2005)