On the Security Validation of Integrated Security Solutions*

Andreas Fuchs, Sigrid Gürgens, and Carsten Rudolph

Fraunhofer Institute for Secure Information Technology
{andreas.fuchs,sigrid.guergens,carsten.rudolph}@sit.fraunhofer.de

Abstract. Combining security solutions in order to achieve stronger (combined) security properties is not straightforward. This paper shows that security-preserving alphabetic language homomorphisms can be used to derive security results for combined security solutions. A relatively simple example of the combination of two different authentication properties (device authentication using a trusted platform module and user authentication using SSL) are integrated. Using security-preserving language homomorphisms it is shown that previously proposed combinations of solutions do not satisfy the desired integrated security properties. Finally, an improved integration of the two solutions is shown to satisfy the desired properties.

1 Introduction

Complex security properties can often not be realised by one single security mechanism in which case it is necessary to combine several mechanisms. This combination is usually far from trivial for several reasons. First, the complex security property is often different (stronger) than the union of the properties of the separate solutions. Furthermore, solutions need to be integrated in the correct way. This paper provides a relatively small example of two integrated security solutions where one intuitive approach does not satisfy the desired security properties. The security goal in this example is the combination of client authentication by using a secure channel, namely SSL, with the identification of the end-points of the channel based on TPM attestation. A formal approach based on formal languages and security-preserving alphabetic language homomorphisms is used to prove the security properties of the integration. Remarkably, this formal approach shows that two existing proposals for integrating TPM attestation with secure channel establishment [1,5] do not satisfy the desired combined security property.

Formal methods have been extensively studied and used for the security analysis of single security mechanisms like cryptographic protocols (see for example [13,12,9]). However, combinations of security solutions have mainly been studied

* Part of this work was accomplished within the project SERENITY 27587 funded by the European Commission.

D. Gritzalis and J. Lopez (Eds.): SEC 2009, IFIP AICT 297, pp. 190–201, 2009.

in the context of additional attack possibilities derived from combining protocols [2,11] and refinement for security properties, in particular non-repudiation [10,14]. In contrast to these approaches, this paper shows a pragmatic way of applying formal methods to verify the security properties of integrated security solutions. This approach is based on an existing framework for security property specification [7,8]. This framework is briefly revisited in the following section before discussing the example of integrating SSL channels with TPM attestation.

2 The Underlying Framework for Security Property Specification

In this section we first give a very brief summary of the necessary concepts of formal languages to describe system behaviour and abstractions.

The behaviour B of a discrete system can be formally described by the set of its possible sequences of actions (traces). Therefore $B \subseteq \Sigma^*$ holds where 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 [3].

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

We further extend the system specification by two components: *agents' initial knowledge* about the global system behaviour and *agents' view*. 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. Further, 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 can use its *local view* of ω (denoted by λ) to determine the sequences of actions it considers to be possible. For a sequence of actions $\omega \in B$ and agent $P \in \mathbb{P}$ (where $\mathbb{P}$ denotes 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 B, underlying security mechanisms and system assumptions, P does not consider all sequences in $\lambda_P^{-1}(\lambda_P(\omega))$ possible. Thus it can use its knowledge to reduce this set: $\lambda_P^{-1}(\lambda_P(\omega)) \cap W_P$ describes all sequences of actions P considers to be possible when ω has happened.

Security properties can now be defined in terms of the agents' initial knowledge and local views. For more details we refer the reader to [7].

Our definition of authenticity (see [7]) uses the above described concepts and essentially states that a set of actions $\Gamma \subseteq \Sigma$ is authentic for agent P if in all sequences that P considers possible after a sequence of actions ω has happened, some time in the past an action of Γ must have happened.

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

The following definition (see again [7]), specifies sufficient conditions for a homomorphism to preserve authenticity.

Definition 2. *Let $h : \Sigma^* \to \Sigma'^*$ be an alphabetic language homomorphism and for $P \in \mathbb{P}$ let $\lambda_P : \Sigma^* \to \Sigma_P^*$ and $\lambda'_P : \Sigma'^* \to \Sigma_P'^*$ be the homomorphisms describing the* local views *of P on Σ and Σ', respectively. The language homomorphism h preserves authenticity on B if for each $P \in \mathbb{P}$ exists a mapping $h'_P : \lambda_P(B) \to \lambda'_P(B')$ with $\lambda'_P \circ h = h'_P \circ \lambda_P$ on B.*

$f \circ g$ denotes the composition of functions f and g, while Σ_P^* and $\Sigma_P'^*$ denote the images of the respective local views (the actual sets can only be determined for concrete local views).

Finally the next theorem provides the link between authenticity properties of systems on different levels of abstraction (see [7] for more details and for the proof of the theorem).

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

3 Example: Integration of Two Security Solutions

3.1 The System Model

The system that we consider here consists of a server S, two clients C_1, C_2, two devices d_1, d_2, and an arbitrary number of channels ch_j, $j \in \mathbb{N}$. The devices and the server are connected via some kind of network and the clients may use any of these devices to send messages to the server. In this notion, the clients are human beings or applications, whilst the server and the entities denoted as devices are computer systems.

The system shall meet two security requirements: Messages of a client to the server shall be authentic for the server, and at the same time the server shall be able to identify the device the message was sent from. For each of these requirements there are standard solutions available.

3.1.1 User Authentication

The chosen scheme for user authentication is a simplified version of an authentic channel establishment as provided by SSL [4]. It is well-known that SSL with client certificate and signature can be used to provide authenticity of the client. Security analyses of SSL or of particular SSL implementations are not considered in this paper. We assume that SSL indeed establishes a secure authentic channel.

An abstract system can be modeled by introducing two actions: *ssl-init*(C_i, $ch_j(S)$) models the initiation of the SSL handshake with server S by one of the clients C_i on channel ch_j, and *ssl-rec*($S, ch_j(C_i)$) models the completion of the SSL handshake by the server which establishes channel ch_j. Although the reduction of the SSL model to these two actions presents a considerable abstraction of the complex nature of the SSL session key establishment (e.g. we do not model the freshness of the channel) it is sufficient for our purposes.

This system provides the property that each time the server performs *ssl-rec* for a channel $ch_j(C_i)$, the handshake initialisation *ssl-init*($C_i, ch_j(S)$) by client C_i is authentic for the server as defined in Definition 1.

3.1.2 Device Identification

The solution that we choose for device identification is based on trusted computing technology as specified by the Trusted Computing Group [6]. The Trusted Platform Module (TPM) can be used to attest the integrity of software running on the platform it is integrated in. For this the software is measured (hashed) and the resulting value is stored in so-called Platform Configuration Registers (PCR) which are only accessible by the TPM. Then the TPM_Quote command instructs the TPM to calculate a signature over these PCR values. By associating the signature key with the TPM, the result of the TPM_Quote can be used to identify the platform. This description omits many details of the very complex process, for more information see [15].

A very abstract model of this solution uses three actions: Action *att-gen*(d_k, *quote*(d_k)) models the generation of *quote*(d_k) (the signature on the PCR values) using device d_k, *att-send*(d_k, S, *quote*(d_k)) models the sending of the quote message to the server, and *att-rec*(S, *quote*(d_k)) models the reception of the quote message by the server. This system provides the property that each time the server performs *att-rec* for a *quote*(d_k) message, the generation of this message by device d_k is authentic for the server as defined in Definition 1.

3.2 Specification of the Abstract Integrated System

We will now model an idealized abstract system behaviour B that provides both user authenticity and device identification simultaneously by defining the following actions in Σ:

- *ssl-init*($C_i, ch_j(S), d_k$) models the initiation of the SSL handshake with server S by one of the clients C_i on channel ch_j using one of the devices d_k.
- *ssl-rec*($S, ch_j(C_i, d_k)$) models the completion of the SSL handshake by the server which establishes channel ch_j. The parameters C_i and d_k of the channel

denote that server S considers the channel to be initiated by client C_i and that the end-point of the channel should be d_k.

The abstract system shall satisfy the property that every time the server completes an SSL handshake for a channel $ch_j(C_i, d_k)$, the channel was indeed initiated by client C_i using device d_k. This can be formalized as follows:

Property 1. $\forall \omega \in B$ *holds if* ssl-rec$(S, ch_j(C_i, d_k)) \in alph(\omega)$ *then* ssl-init$(C_i, ch_j(S), d_k)$ *is authentic for* S.

For the abstract model we assume the existence of a security mechanism that simultaneously authenticates client and device during establishment of a channel. Since for expressing this property we only need the view of server S, we define the assumptions for the system behaviour B in terms of the initial knowledge of the server:

$$B = \Sigma^* \setminus (W_S^1 \cup W_S^2 \cup W_S^3)$$

where W_S^1, W_S^2 and W_S^3 describe those sequences of actions that violate the properties we want the abstract system to provide:

$$W_S^1 = \bigcup_{j \in \mathbb{N}, i,k,l \in \{1,2\}} (\Sigma \setminus \{\mathit{ssl\text{-}init}(C_i, ch_j(S), d_k)\})^* \{\mathit{ssl\text{-}rec}(S, ch_j(C_i, d_l)\}^* \ \Sigma^*$$

describes that each handshake performed by the server on channel $ch_j(C_i,\ d_l)$ corresponds to a handshake initiation by the same client C_i on the same channel ch_j on some device d_k.

$$W_S^2 = \bigcup_{j \in \mathbb{N}, i,k,l,m \in \{1,2\}} \Sigma^* \{\mathit{ssl\text{-}init}(C_i, ch_j(S), d_k)\}\ \Sigma^* \{\mathit{ssl\text{-}init}(C_m, S, ch_j(S), d_l)\}\ \Sigma^*$$

describes that a channel can only be initiated once.

Finally, we assume that the server S recognizes the device on which the client started the handshake:

$$W_S^3 = \bigcup_{j \in \mathbb{N}, i,k,l \in \{1,2\}} (\Sigma \setminus \{\mathit{ssl\text{-}init}(C_i, ch_j(S), d_k)\})^* \{\mathit{ssl\text{-}rec}(S, ch_j(C_l, d_k)\}^* \ \Sigma^*$$

describes that each time the server performs a handshake that was presumably initiated on device d_k, some client indeed initiated the handshake on this device. In other words, the device that was used to initiate the handshake is authentic for the server.

It is easy to show that this system provides Property 1.

3.3 Specification of the Concrete Integrated System

3.3.1 A Naive Integration

The two mechanisms described in Sections 3.1.1 and 3.1.2 connect the client to a channel and the quote message to a device, respectively. By integrating

these mechanisms one would expect to achieve a link between client, channel and device simultaneously. The most obvious approach of such an integration is to first start up an SSL-connection and then send the result of a TPM_Quote Attestation over that channel. In the following we will show that this approach does not satisfy the desired properties.

To this end we define a concrete system $\boldsymbol{B}$. It has the same participants C_1, C_2 and S, and uses the same channels $ch_j, j \in \mathbb{N}$, and devices d_1, d_2 as does the abstract system but uses a set of refined actions.

In the following, we use italic font for the abstract system and actions while using italic boldface font for the concrete system and actions.

Thus $\boldsymbol{\Sigma}$ contains the following actions:

- $\boldsymbol{ssl\text{-}init}(C_i, ch_j(C_i, S), d_k)$ As in the abstract system, this action models the initiation of the SSL handshake on channel ch_j by one of the clients C_i, using one of the devices. The channel has both endpoints as parameters.
- $\boldsymbol{ssl\text{-}rec}(S, ch_j(C_i, S))$ models the completion of the SSL handshake by the server which establishes channel $ch_j(C_i, S)$. Note that here S cannot tell to which device the end-point of the channel is connected. This information shall be provided in the subsequent attestation.
- $\boldsymbol{att\text{-}gen}(d_i, quote(d_k))$ This action models the generation of $quote(d_k)$ using device d_i.
- $\boldsymbol{att\text{-}send}(C_i, quote(d_k), ch_j(C_i, S))$ This action models the sending of the attestation message $quote(d_k)$ on channel $ch_j(C_i, S)$.
- $\boldsymbol{att\text{-}rec}(S, quote(d_k), ch_j(C_i, S))$ models the reception of the attestation message by the server on channel $ch_j(C_i, S)$.

For the refined system $\boldsymbol{B}$ we assume that SSL and TPM attestation are used to provide authenticity of the client and device identification, respectively. Thus, the system behaviour can be restricted through adequate assumptions representing the assumed properties for these solutions. Thus, we construct the system behaviour $\boldsymbol{B}$ based on the initial knowledge of the server S in the concrete system as $\boldsymbol{B} = \boldsymbol{\Sigma}^* \setminus (\boldsymbol{W_S^1} \cup \boldsymbol{W_S^2} \cup \boldsymbol{W_S^3} \cup \boldsymbol{W_S^4} \cup \boldsymbol{W_S^5})$ where the $\boldsymbol{W_S^i}$ describe those sequences of actions that violate the properties we assume the concrete system to provide:

Assumption 1. *In analogy to the abstract system B we assume that the two SSL actions provide authenticity of the client to the server. It seems reasonable to assume that in combining the two solutions no SSL keying information is revealed by the TPM attestation process.*

$$\boldsymbol{W_S^1} = \bigcup_{j \in \mathbb{N}, i,k \in \{1,2\}} (\boldsymbol{\Sigma} \setminus \{\boldsymbol{ssl\text{-}init}(C_i, ch_j(C_i, S), d_k)\})^* \{\boldsymbol{ssl\text{-}rec}(S, ch_j(C_i, S)\} \; \boldsymbol{\Sigma}^*$$

Assumption 2. *For the two SSL actions we can also assume that a channel is only established once. This is justified by the fact that both communication parties involved in an SSL-communication influence the session secret and even if only one (the server S) uses a reliable random number generator, the session secret will be virtually unique to this communication session.*

$$\boldsymbol{W}^{\mathbf{2}}_{\mathbf{S}} = \bigcup_{j\in I\!N,i,k,l\in\{1,2\}} (\boldsymbol{\Sigma}^*\{\textbf{\textit{ssl-init}}(C_i, ch_j(C_i,S), d_k)\}\ \boldsymbol{\Sigma}^*\{\textbf{\textit{ssl-init}}(C_i, S, ch_j(C_i,S), d_l)\ \boldsymbol{\Sigma}^*$$

Assumption 3. *By the nature of an SSL channel, if some message is received on it, there must be a respective send action on this channel. In analogy to Assumption 1 we further assume that this send action is authentic. Hence we assume that whenever the server receives an attestation message* $quote(d_k)$ *on channel* $ch_j(C_i, S)$*, the message must have been sent on this channel by the client* C_i*.*

$$\boldsymbol{W}^{\mathbf{3}}_{\mathbf{S}} = \bigcup_{j\in I\!N,i,k\in\{1,2\}} (\boldsymbol{\Sigma} \setminus \{\textbf{\textit{att-send}}(C_i, quote(d_k), ch_j(C_i,S))\})^*\{\textbf{\textit{att-rec}}(S, quote(d_k), ch_j(C_i,S))\}\ \boldsymbol{\Sigma}^*$$

Assumption 4. *We assume that the three attestation actions provide authenticity of the device for the server. Each time the server receives an attestation message* $quote(d_k)$ *on some channel* $ch_j(C_i, S)$*, in all sequences it considers possible indeed this device generated the message.*

$$\boldsymbol{W}^{\mathbf{4}}_{\mathbf{S}} = \bigcup_{j\in I\!N,i,k\in\{1,2\}} (\boldsymbol{\Sigma} \setminus \{\textbf{\textit{att-gen}}(d_k, quote(d_k))\})^*\{\textbf{\textit{att-rec}}(S, quote(d_k), ch_j(C_i,S))\}\boldsymbol{\Sigma}^*$$

Assumption 5. *SSL channels can only be used after a successful handshake. Thus, whenever a client sends the attestation message on channel* $ch_j(C_i, S)$*, the server has established this channel before.*

$$\boldsymbol{W}^{\mathbf{5}}_{\mathbf{S}} = \bigcup_{j\in I\!N,i,k\in\{1,2\}} (\boldsymbol{\Sigma} \setminus \{\textbf{\textit{ssl-rec}}(S, ch_j(C_i,S))\})^*\{\textbf{\textit{att-send}}(C_i, quote(d_k), ch_j(C_i,S))\}\boldsymbol{\Sigma}^*$$

4 Security Validation

In this section we will show that the concrete system defined in the previous section does not provide simultaneous authenticity of client and identification of device. We do this by formulating a proposition to Theorem 1 and trying to prove it using security preserving language homomorphisms as explained in Section 2.

Proposition 1. *For all* $\omega \in \boldsymbol{B}$ *with* $\textbf{\textit{att-rec}}(S, quote(d_k), ch_j(C_i,S)) \in alph(\omega)$ *holds that* $\textbf{\textit{ssl-init}}(C_i, S, ch_j(C_i,S), d_k)$ *is authentic for* S *after* ω*.*

In order to prove this proposition we need to find a language homomorphism that maps the concrete system $\boldsymbol{B}$ to the abstract system B and preserves authenticity. Then we can conclude on the authenticity properties that hold in the refined system.

4.1 Local Views and a Possible Homomorphism

Since we are interested in a security property concerning the server we only define its respective local views and disregard those of the clients. In a distributed system it is appropriate to assume that S can only see its own actions:

$$\lambda_{\mathbf{S}}(\mathbf{a}) = \begin{cases} a \text{ if } a \in \mathbf{\Sigma}_{/\mathbf{S}} \\ \varepsilon \ else \end{cases} \qquad \lambda_S(a) = \begin{cases} a \text{ if } a \in \Sigma_{/S} \\ \varepsilon \ else \end{cases}$$

We now define homomorphism h to relate the two systems. In the concrete model, ***att-rec*** shall establish the binding of the channel to the device. Thus, this action is mapped to *ssl-rec* in the abstract system.

$$\begin{aligned} h(\textbf{\textit{ssl-init}}(C_i, ch_j(C_i, S), d_k)) &= \textit{ssl-init}(C_i, ch_j(S), d_k) \\ h(\textbf{\textit{att-rec}}(S, quote(d_k), ch_j(C_i, S))) &= \textit{ssl-rec}(S, ch_j(C_i, d_k)) \\ h(\textbf{\textit{ssl-rec}}(S, ch_j(C_i, S))) &= \varepsilon \\ h(\textbf{\textit{att-gen}}(d_k, quote(d_k))) &= \varepsilon \\ h(\textbf{\textit{att-send}}(C_i, quote(d_k), ch_j(C_i, S))) &= \varepsilon \end{aligned}$$

4.2 Proof Attempt

According to Theorem 1, in order to prove that h preserves authenticity we need to find a homomorphism $h'_S : \lambda_{\mathbf{S}}(\boldsymbol{B}) \rightarrow \lambda_S(B)$ that is compliant both with h and the local views of the server in the abstract and concrete system, respectively. It is easy to see that the homomorphism defined in the following has this property:

$$\begin{aligned} h'_S(\textbf{\textit{ssl-rec}}(S, ch_j(C_i, S))) &= \varepsilon \\ h'_S(\textbf{\textit{att-rec}}(S, quote(d_k), ch_j(C_i, S))) &= \textit{ssl-rec}(S, ch_j(C_i, d_k)) \end{aligned}$$

So in order to be able to apply Theorem 1 we need to show that $h(\boldsymbol{B}) \subseteq B$.

Proof 1. *We show the reverse, namely that for $\omega \notin B$, i.e. for $\omega \in W_S^1 \cup W_S^2 \cup W_S^3$, it follows for all $x \in h^{-1}(\omega)$ that $x \notin \boldsymbol{B}$. For $\omega \in W_S^1$, the assertion can easily be shown using Assumptions 3, 5 and 1, for $\omega \in W_S^2$ the assertion follows from Assumption 2.*

The interesting case is $\omega \in W_S^3$. Then ω contains ssl-rec$(S, ch_j(C_i,\ d_k))$ *with no action* ssl-init$(C_l, ch_j(S), d_k)$ *before. For $x \in h^{-1}(\omega)$ it follows that x contains* ***att-rec***$(S, quote(d_k), ch_j(C_i, S))$*. With Assumption 3 we can conclude that there is an action* ***att-send***$(C_i, quote(d_k), ch_j(C_i, S))$ *before. Assumption 5 allows to conclude* ***ssl-rec***$(S, ch_j(C_i, S)) \in alph(x)$ *and Assumption 1 leads to* ***ssl-init***$(C_i, ch_j(C_i, S), d_l) \in alph(x)$*. Since h maps this action onto* ssl-init$(C_i, ch_j(S), d_l)$ *and $x \in h^{-1}(\omega)$ and $\omega \in W_S^3$, it follows $d_l \neq d_k$. In order to show $x \notin \boldsymbol{B}$ we need an assumption that refers to the devices being used in the actions. However, the only such assumption we have available is Assumption 4 which only allows to conclude that the generation of the message $quote(d_k)$ indeed happened on device d_k. We cannot prove $x \notin \boldsymbol{B}$ because there is no link between* ***att-gen***$(d_k, quote(d_k))$ *and the channel used in* ***att-send***$(C_i, quote(d_k), ch_j(C_i, S))$*.*

Failing to show that the mapping defined in Section 4.1 indeed maps the concrete system $\boldsymbol{B}$ onto the abstract system B indicates that the concrete system might after all not provide the desired property.

4.3 Analysis of the Proof Attempt

Taking the failure of the proof as an input for a manual security evaluation indicates that the device establishing the handshake does not necessarily have to be the device that generates the quote message. The failed proof identifies a counter example: $\boldsymbol{ssl\text{-}init}(C_i, ch_j(C_i, S), d_l)\boldsymbol{ssl\text{-}rec}(S, ch_j(C_i, S))\boldsymbol{att\text{-}gen}(d_k, quote(d_k))$ $\boldsymbol{att\text{-}send}(C_i, quote(d_k), ch_j(C_i, S))$ $\boldsymbol{att\text{-}rec}(S, quote(d_k), ch_j(C_i, S))$ is one of the sequences of actions that h maps onto $\omega \notin B$ but that do not violate any of the assumptions for $\boldsymbol{B}$.

A possible attack scenario that exploits the missing link between the handshake initialization and the quoting device is the following: A company policy restricts access to its server to on-site or home offices, but disallows mobile access with laptops from trains or internet-cafes. A malicious client wants to connect to the server from e.g. a train. He uses SSL and e.g. a smart-card to authenticate himself. Then he establishes a connection to the home office PC, generates a quote message by this PC, and sends the quote message back to the server using the SSL channel. Even if the quote verification did not allow manual quote calls, it is still possible to use dns-spoofing for a man-in-the-middle attack and proceed similarly.

4.4 A Fixed Concrete System

In order to formally add the missing link between the device and the channel (in the next section we will discuss possible realizations) we add the channel to the quote message. We further assume that the channel contained in a quote message is always connected to the device that produced this message. Thus we obtain the following changes to the attestation actions:

- $\boldsymbol{att\text{-}gen}(d_k, quote(d_k, ch_j(C_i, S)))$
- $\boldsymbol{att\text{-}send}(C_i, quote(d_k.ch_j(C_i, S)), ch_j(C_i, S))$
- $\boldsymbol{att\text{-}rec}(S, quote(d_k, ch_j(C_i, S)), ch_j(C_i, S))$

Assumption 6. *The channel that is contained in the quote message is initiated on the device that generates this message. Thus the sequences specified below can not be part of the concrete system:*

$$\boldsymbol{W}^{\mathbf{6}}_{\mathbf{S}} = \bigcup_{j \in \mathbb{N}, i,k \in \{1,2\}} (\boldsymbol{\Sigma} \setminus \{\boldsymbol{ssl\text{-}init}(C_i, ch_j(C_i, S), d_k)\})^* \{\boldsymbol{att\text{-}gen}(d_k, quote(d_k, ch_j(C_i, S))\} \boldsymbol{\Sigma}^*$$

Further assumptions regarding the chronology of $\boldsymbol{att\text{-}gen}$ can be made but are not necessary for the proof.

Our fixed system $\boldsymbol{B}_{\mathbf{fix}}$ is now defined as

$$\boldsymbol{B}_{\mathbf{fix}} = \boldsymbol{\Sigma}^* \setminus (\boldsymbol{W}^{\mathbf{1}}_{\mathbf{S}} \cup \boldsymbol{W}^{\mathbf{2}}_{\mathbf{S}} \cup \boldsymbol{W}^{\mathbf{3}}_{\mathbf{S}} \cup \boldsymbol{W}^{\mathbf{4}}_{\mathbf{S}} \cup \boldsymbol{W}^{\mathbf{5}}_{\mathbf{S}} \cup \boldsymbol{W}^{\mathbf{6}}_{\mathbf{S}})$$

Using these assumptions we can now prove that the concrete fixed system simultaneously provides client authentication and device identification. We use the homomorphism h_{fix} equivalent to the one used for the flawed

system that maps $\textbf{\textit{ssl-init}}(C_i, ch_j(C_i, S), d_k)$ onto $\textit{ssl-init}(C_i, ch_j(S), d_k)$ and $\textbf{\textit{att-rec}}(S, quote(d_k, ch_j(C_i, S)), ch_j(C_i, S))$ onto $\textit{ssl-rec}(S, ch_j(C_i, d_k))$, and all other actions onto the empty word.

Clearly, also this homomorphism preserves authenticity. It remains to show that $h_{fix}(\boldsymbol{B}_{\mathbf{fix}}) \subseteq B$. Again, we will show that for $\omega \notin B$, all $x \in h_{fix}^{-1}(\omega)$ are not elements of $\boldsymbol{B}_{\mathbf{fix}}$.

Proof 2. *The cases $\omega \in W_S^1$ and $\omega \in W_S^2$ are analogous to the proof attempt presented in Section 4.2. The interesting case is the one were our first proof failed:*

Let $\omega \in W_S^3$. As in the previous proof, Assumptions 3, 5, and 1 imply the existence of a sequence $x \in h_{fix}^{-1}(\omega)$ with actions $\textbf{\textit{ssl-init}}(C_i, ch_j(C_i, S), d_l)$, $\textbf{\textit{ssl-rec}}(S, ch_j(C_i, S))$, $\textbf{\textit{att-send}}(C_i, quote(d_k, ch_j(C_i, S)), ch_j(C_i, S))$, and $\textbf{\textit{att-rec}}(S, quote(d_k, ch_j(C_i, S)), ch_j(C_i, S))$ in this order. Furthermore, from $\textbf{\textit{att-rec}}(S, quote(d_k, ch_j(C_i, S)), ch_j(C_i, S)) \in alph(x)$ we can conclude, using Assumption 4, that $\textbf{\textit{att-gen}}(d_k, quote(d_k, ch_j(C_i, S)))$ must have happened before, and Assumption 6 implies that $\textbf{\textit{ssl-init}}(C_i, ch_j(C_i, S), d_k)$ must have happened before $\textbf{\textit{att-gen}}(d_k, quote(d_k, ch_j(C_i, S)))$. Finally Assumption 2 implies $d_l = d_k$. By the definition of h_{fix} this implies ssl-init$(C_i, ch_j(S), d_k) \in \omega$, *a contradiction to the assumption we started with. Hence one of the assumptions for the concrete system is violated and thus $x \notin \boldsymbol{B}_{\mathbf{fix}}$.*

The proof also shows that the quote message does not need to be sent on the SSL-Channel because the proof still holds with a respectively altered Assumption 4, as the link between the channel and the device results from the quote generation. However confidentiality concerns may imply to use the SSL channel nonetheless.

5 Practical Realisation of a Secure Integration

A possible realisation for the fixed system $\boldsymbol{B}_{\mathbf{fix}}$ could be to reserve a PCR on the TPM for the sole purpose of authenticating the device's connections: Whenever an SSL connection is being established, the device would save the handshake messages into this PCR. The attested chain of integrity measurement values could prove that the platform will only extend the PCR by handshake messages for SSL sessions on this device. A local daemon on the platform could control the measurement of SSL session establishment on the platform.

Previously proposed approaches to construct a secure remote attestation fail to satisfy the non-obvious Assumption 6. In [5] a PCR dedicated to store the SSL client Public Key or Certificate and an additional Platform Property Certificate is proposed. This fails if the SSL private key is compromised and the client is tricked into providing a TPM_Quote. More importantly, it also fails if the client wants to pretend the use of a different device. In this case Assumption 6 is not justified since the client's certificate can very well be available on more than one device. The Network Interface Monitoring Agent (NIMA) introduced in [1] links the channel endpoint in terms of the IP-Address to the TPM_Quote by storing it in a PCR. This will also fail because a second platform could pretend to have the IP-Address of the attested platform.

6 Conclusions

In this paper we have shown that the combination of formal security protocol specifications with security-preserving language homomorphisms can indeed provide new insight into the properties of integrated security solutions. Although we used a very simplified model of the integration of TPM-based attestation with SSL security channels we were able to show that previously proposed integrations do not provide the desired security properties while a more sophisticated integration does. The example in this paper addresses only two very similar security properties. However, our framework allows to handle other important security properties such as different instantiations of confidentiality and non-repudiation.

Our method emphazises the importance of assumptions made on a particular system. While the assumptions used to prove the desired properties are reasonable with respect to the practical realisation of the integration we discussed, any further assumption for the trivial system that would allow a proof can not be argued.

Furthermore, the assumptions used in a proof can be the basis for design-time or run-time monitoring checks to verify their justification. This provides information about the applicability of an integrated security solution.

References

1. Choi, S., Han, J., Jun, S.: Improvement on TCG Attestation and Its Implication for DRM. In: Gervasi, O., Gavrilova, M.L. (eds.) ICCSA 2007, Part I. LNCS, vol. 4705, pp. 912–925. Springer, Heidelberg (2007)
2. Cremers, C.: Feasibility of multi-protocol attacks. In: Proc. of The First International Conference on Availability, Reliability and Security (ARES), pp. 287–294. IEEE Computer Society, Los Alamitos (2006)
3. Eilenberg, S.: Automata, Languages and Machines. Academic Press, London (1974)
4. Frier, A., Karlton, P., Kocher, P.: The SSL 3.0 Protocol. Netscape Communications Corp. (November 1996)
5. Goldman, K., Perez, R., Sailer, R.: Linking remote attestation to secure tunnel endpoints. In: STC 2006: Proceedings of the first ACM workshop on Scalable trusted computing, pp. 21–24. ACM, New York (2006)
6. Trusted Computing Group. TCG TPM Specification 1.2 revision 94 (2006), `http://www.trustedcomputing.org`
7. Gürgens, S., Ochsenschläger, P., Rudolph, C.: Authenticity and provability - A formal framework. In: Davida, G.I., Frankel, Y., Rees, O. (eds.) InfraSec 2002. LNCS, vol. 2437, pp. 227–245. Springer, Heidelberg (2002)
8. 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), 457–466 (2005)
9. Gürgens, S., Rudolph, C., Scheuermann, D., Atts, M., Plaga, R.: Security evaluation of scenarios based on the tCG's TPM specification. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 438–453. Springer, Heidelberg (2007)

10. Mantel, H.: Preserving Information Flow Properties under Refinement. In: IEEE Symposium on Security and Privacy, Oakland, pp. 78–91. IEEE Computer Science, Los Alamitos (2001)
11. Mathuria, A., Singh, A., Shravan, P.V., Kirtanka, R.: Some new multi-protocol attacks. In: ADCOM – Proceedings of the 15th International Conference on Advanced Computing and Communications, pp. 465–471. IEEE Computer Society, Los Alamitos (2007)
12. Meadows, C.: Analyzing the Needham-Schroeder Public Key Protocol: A Comparison of Two Approaches. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146. Springer, Heidelberg (1996)
13. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security 6, 85–128 (1998)
14. Santen, T.: Preservation of probabilistic information flow under refinement. Information and Computation 206(2-4), 213–249 (2008)
15. Trusted Computing Group. TPM Main - Part 1 Design Principals, Specification Version 1.2, Level 2 Revision 103 (July 2007)