Preserving Confidentiality in Component Compositions

Andreas Fuchs, Sigrid Gürgens

Preserving Confidentiality in Component Compositions

Andreas Fuchs and Sigrid Gürgens

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

Abstract. The preservation of any security property for the composition of components in software engineering is typically regarded a non-trivial issue. Amongst the different possible properties, confidentiality however poses the most challenging one. The naive approach of assuming that confidentiality of a composition is satisfied if it is provided by the individual components may lead to insecure systems as specific aspects of one component may have undesired effects on others. In this paper we investigate the composition of components that each on its own provide confidentiality of their data. We carve out that the complete behaviour between components needs to be considered, rather than focussing only on the single interaction points or the set of actions containing the confidential data. Our formal investigation reveals different possibilities for testing of correct compositions of components, for the coordinated distributed creation of composable components, and for the design of generally composable interfaces, ensuring the confidentiality of the composition.

1 Introduction

Software design and engineering makes strong use of composition in many ways. From the orchestration of web services in a Business Process Engines to the integration of libraries or object files compilers and linkers the principles of composition apply on any of these layers of abstraction.

Beyond the general problems of feature interaction, there exist many specific security related challenges that can introduce serious flaws in a software product. A prominent example for such a flaw is the integration of TLS libraries into the German eID Application [1] that caused the acceptance of update packages by any server with a valid certificate, as the name within the certificate was not checked. In other cases, integrators of TLS libraries do not provide enough entropy for key generation which leads to a series of servers on the Internet with similar private key values [2].

Practical solutions for composition include the provisioning of verbal best-practice catalogues [3], tool-based solution databases [4–6] or guides, tutorials and code examples in general. However, there does not exist much research that targets the challenges imposed by composition on a more general and broader scope.

In this contribution we present an approach based on the formal semantics of our Security Modelling Framework SeMF (see e.g. [7–9]) that targets the investigation and validation of general component composition regarding the property of data confidentiality. SeMF has available a comprehensive vocabulary for statements of confidentiality that provides the necessary expressiveness to reason about conditions of general composability decoupled from any specific scenario.

In the following Section we introduce a scenario that serves as test case for our approach. It is composed of two components that (each on its own) provide a certain confidentiality property, but fail to do so when composed into a joined system. Section 3 gives a brief introduction to the SeMF framework. In Section 4 we introduce our formalization of system composition and demonstrate it using the example scenario. We then explain and formalize the conditions for confidentiality composition in Section 5 and illustrate them by means of the example scenario in Section 6. Section 7 provides an overview of related work on composition of security and Section 8 finalizes the paper and provides an outlook to ongoing and future work.

2 Example Scenario

The provisioning and quality of entropy is a central aspect for many security functionalities. However, the generation of entropy and randomness in computers is a hard problem on its own [10] and at the same time programmers are usually not introduced to its challenges in a correct way. Many code examples and explanations for randomness today advice to use the current time or uptime as seed for a random number generator. This approach may be adequate for desktop applications started by the user at an unforeseeable as well as undetectable point in time. Whenever these conditions do not hold however – as is the case especially in e.g. system service applications or embedded platforms [11] – such date/uptime values do not provide enough entropy. These scenarios rather require specialized entropy sources in CPU through a TPM or a SmartCard.

In our example scenario, we investigate such a case, i.e. a system which is composed of a security library for key generation that targets desktop applications whilst being utilized by an embedded platform system service. The $KeyGenerator$ component hereby uses the current time of the system when being called in order to initialize its random number generator and to create the corresponding key. The $Application$ component of the system represents a system service that is started during boot and calls the $KeyGenerator$ for a key to be generated. Both components have the property of confidentiality for the key that is generated / further used. However, their composition introduces side effects that make the key calculable for a third party.

3 Formal Semantics of SeMF

In our Security modelling Framework SeMF, the specification of any kind of cooperating system is composed of (i) a set $\mathbb{P}$ of agents (e.g. an application and a key generator), (ii) a set Σ of actions, (iii) the system's behaviour $B \subseteq \Sigma^*$ (Σ^* denoting the set of all words composed of elements in Σ), (iv) the local views $\lambda_P : \Sigma^* \to \Sigma_P^*$, and (v) initial knowledge $W_P \subseteq \Sigma^*$ of agents $P \in \mathbb{P}$. The behaviour B of a discrete system S can be formally described by the set of its possible sequences of actions (which is always prefix closed). An agent P's initial knowledge W_P about the system consists of all traces the agent initially considers possible. This includes a representation of conclusions that an agent may be able to derive; i.e. that the reception of a message implies the sending of this message to have happened before. Finally, an agent's local view essentially captures what an agent can see from the system. Together, the local view and initial knowledge

represent what an agent may know about the system at a given point in time based on what he/she knows in general, has seen and has concluded from this. Different formal models of the same system are partially ordered with respect to the level of abstraction. Formally, abstractions are described by alphabetic language homomorphisms that map action sequences of a finer abstraction level to action sequences of a more abstract level while respecting concatenation of actions. In fact, the agents' local views are expressed by homomorphisms. Note that homomorphisms are in general neither injective nor surjective. For $\Sigma_1 \subseteq \Sigma_2$, the homomorphism $h : \Sigma_2 \longrightarrow \Sigma_1$ that keeps all actions of Σ_1 and maps those in $\Sigma_2 \setminus \Sigma_1$ onto the empty word is called *projection homomorphism*.

In SeMF, security properties are defined in terms of such a system specification. Note that system specification does not require a particular level of abstraction. The underlying formal semantics then allows formal validation, i.e. allows to prove that a specific formal model of a system provides specific security properties.

3.1 Confidentiality in SeMF

Based on the SeMF semantics, we have specified various instantiations of security properties such as precedence, integrity, authenticity and trust (see e.g. [7, 12, 13]). In this paper however we focus on our notion of parameter confidentiality [8, 9]. Various aspects are included in this concept. First, we have to consider an attacker Eve's local view λ_{Eve} of the sequence ω she has monitored and thus the set of sequences $\lambda^{-1}_{Eve}(\lambda_{Eve}(\omega))$ that are, from Eve's view, identical to ω. Second, Eve can discard some of the sequences from this set, depending on her knowledge of the system and the system assumptions, all formalized in W_{Eve}. For example, there may exist interdependencies between the parameter p to be confidential in different actions, such as a credit card number remaining the same for a long time, in which case Eve considers only those sequences of actions possible in which an agent always uses the same credit card number. The set of sequences Eve considers possible after ω is $\lambda^{-1}_{Eve}(\lambda_{Eve}(\omega)) \cap W_{Eve}$. Third, we need to identify the actions in which the respective parameter(s) shall be confidential. Many actions are independent from these and do not influence confidentiality, thus need not be considered. For this we use a homomorphism $\mu : \Sigma^* \longrightarrow (\Sigma_\tau \times M)^*$ that maps actions to be considered onto a tuple $(actiontype, parameter)$.

Essentially, parameter confidentiality is captured by requiring that for the actions that shall be confidential for Eve with respect to some parameter p, all possible (combinations of) values for p occur in the set of actions that Eve considers possible. What are the possible combinations of parameters is the fourth aspect that needs to be specified, as we may want to allow Eve to know some of the interdependencies between parameters (e.g. in some cases Eve may be allowed to know that the credit card number remains the same, in others we may want to require Eve not to know this). The notion of (L, M)–Completeness captures which are the dependencies allowed to be known within a set of sequences of actions. For the formal definition of (L, M)–completeness, some additional notations are 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, and $p_\tau : (\Sigma_t \times M) \longrightarrow \Sigma_t$ is a mapping that removes the parameters.

Definition 1 ((L,M)-completeness) *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 2 (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*

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

Here $p_\tau^{-1} \circ p_\tau$ first removes and then adds again all values of the parameter that shall be confidential, i.e. constructs all possible value combinations. (L, M)–completeness of K captures that R is required to consider all combinations of parameter values possible except for those that it is allowed to disregard (i.e. those that are not in K). Hence the right hand side of the inequality specifies all sequences of actions agent R shall consider as the ones that have possibly happened after ω has happened. In contrast, the left hand side represents those sequences that R actually does consider as those that have possibly happened. For further explanations we refer the reader to [8, 9].
Notation: We will use $\Lambda_R(\omega, W_R) = \lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ as an abbreviation.

4 Modelling Composition

Based on SeMF we now introduce the definition of the composition of two systems with the same set of agents and a shared interface. Applying this definition, we then specify the composition of the scenario application and key generator.

4.1 Formalizing Composition

The idea of our formalization is to interpret the individual components S_1 and S_2 as homomorphic images of the composed system and to express this system in terms of the inverses of the components with respect to the homomorphisms. Figure 1 illustrates the relationship between the systems: Both components S_1 and S_2 are abstractions (i.e. images of homomorphisms h_1 and h_2, respectively) of their composition S_0, while S_1 and S_2 in turn are abstracted (by homomorphisms h_1^{IF} and h_2^{IF}, respectively) onto their joined interface. Agent P's initial knowledge about the composition does only contain those sequences that P considers possible for both S_1 and S_2, hence it is given by the intersection of the inverses of the two homomorphisms. Further, agents' local views for the composed system need to capture what agents can see in both S_1 and S_2. The projections of S_1 and S_2 into the interface system will be of interest for a theorem to be introduced in Section 6.2. In the following we formalize this composition approach.

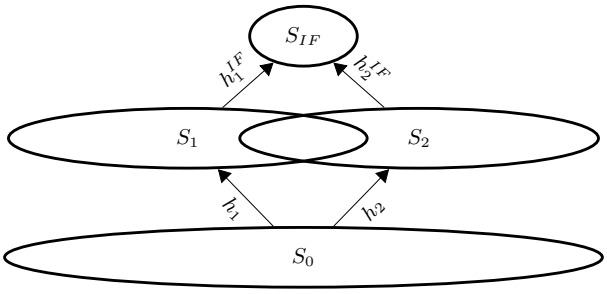

Fig. 1. System relations

Definition 3 (System Composition) *Let S_1 and S_2 be two systems with Σ_i their respective sets of actions, $\mathbb{P}_1 = \mathbb{P}_2$ their set of agents, λ_P^i their agents' local views, and W_P^i their agents' initial knowledge, respectively ($i = 1, 2$). Let further $\Sigma_0 := \Sigma_1 \cup \Sigma_2$, and $h_i : \Sigma_0^* \to \Sigma_i^*$ the projection homomorphisms into Σ_i ($i = 1, 2$). Then the* composition *S_0 of S_1 and S_2 is constructed as follows:*

- $\mathbb{P}_0 := \mathbb{P}_1 = \mathbb{P}_2$,
- $B_0 := h_1^{-1}(B_1) \cap h_2^{-1}(B_2)$
- $W_P^0 := h_1^{-1}(W_P^1) \cap h_2^{-1}(W_P^2)$
- *In order to define the local view of agents in S_0, we define for $i = 1, 2$: $\lambda_P^{i'} : \Sigma_0 \to \Sigma_{P,i}$*
  $\lambda_P^{i'}(a) = \begin{cases} \lambda_P^i(a) \text{ if } a \in \Sigma_i \\ \varepsilon \text{ else} \end{cases}$
  Then the local view of S_0 can be defined as follows:
  $\lambda_P^0(a) := (\lambda_P^{1'}(a), \lambda_P^{2'}(a))$

Further, for $\Sigma_{IF} := \Sigma_1 \cap \Sigma_2$, the projection homomorphisms into Σ_{IF}^* are denoted by $h_i^{IF} : \Sigma_i^* \to \Sigma_{IF}^*$ ($i = 1, 2$).

Note, the above definition is equivalent to $\lambda_P^{1'}(a) = \lambda_P^1(h_1(a))$, $\lambda_P^{2'}(a) = \lambda_P^2(h_2(a))$. Also from the above definition it follows $\Sigma_{0,P} = (\Sigma_{1,P} \times \Sigma_{2,P})$ with $\Sigma_{i,P}$ being the image of λ_P^i ($i = 1, 2$), and $(\lambda_P^0)^{-1}((x, y)) = (\lambda_P^{1'})^{-1}(x) \cap (\lambda_P^{2'})^{-1}(y)$.

4.2 Composing the scenario systems

We now model the interface composition of an application (S_1) and a key generation module (S_2) following the above definition. We assume that the application generates a key directly after each system boot. The model for the application is independant from any key generation modul that is actually being used, and abstracts from the actual key generation (this is not part of the application model and happens magically).
The application model S_1 can be specified as follows:

- Agents of this model (and of S_2) are the application, the key genration module, and a third agent that is not allowed to know the key:
  $\mathbb{P}_1 = \{App, KGen, Eve\}$

– The system is booted, the application calls the key generation module, and the key generation module returns a key $key \in \mathcal{K}$, all actions happening at time $t \in T$:
$\Sigma_1 = \bigcup_{t \in T, key \in \mathcal{K}} \{boot(t), callGenKey(App, t), returnKey(KGen, key, t)\}$
– We assume that Eve can see the time of system boot but can neither see the key generation request nor the key that is returned:
$\lambda^1_{Eve}(boot(t)) = boot(t); \forall a \in \Sigma_1 \setminus \{boot(t) | t \in T\} : \lambda^1_{Eve}(a) = \varepsilon$
– Eve knows that before a key generation request, the system has been booted. For simplicity we assume the period of time between these two actions to be equal to δ_1. Eve may further know that the time of actions in a sequence is strictly monotonic increasing. This is however not relevant for the given scenario. Hence sequences of actions that contradict this fact are not included in Eve's initial knowledge. Formally:
$W^1_{Eve} = \Sigma_1^* \setminus \bigcup_{t_j - t_i = \delta_1} (\Sigma_1 \setminus \{boot(t_i)\})^* \{callGenKey(App, t_j)\} \Sigma_1^*$
– We focus on the confidentiality of the key returned to the application, hence μ_1 maps $returnKey(KGen, key, t_j)$ onto $(returnKey(KGen), key)$ and all other actions onto the empty word.

According to this system model, it is easy to see that the returned key is parameter confidential for Eve regarding μ_1 and *(L,M)-completeness* regarding an adequate L and the set of possible keys M.

We now model a concrete key generation module. This module is not able to retrieve a seed for key generation other than the system clock.

– $\mathbb{P}_2 = \{App, KGen, Eve\}$
– The key generation module is called by the application, generates a key, and returns this key, all actions occurring at a specific time $t \in T$:
$\Sigma_2 = \bigcup_{t \in T, key \in \mathcal{K}} \{callGenKey(App, t), genKey(KGen, key, t), returnKey(KGen, key, t)\}$
– We assume that Eve cannot see any of the actions of the key generation module, hence $\lambda^2_{Eve}(\Sigma_2) = \varepsilon$
– Eve knows that before a key can be generated, the respective key generation call must have happened, and that the time passing between these two actions is at most δ_2. Eve also knows that the key generator only returns keys it has generated before. Eve finally knows that the system time is used as seed for key generation. Formally:

$$W^2_{Eve} = \Sigma_2^* \setminus \bigcup_{t_j - t_i = \delta_2} (\Sigma_2 \setminus \{callGenKey(App, t_i)\})^* \{genKey(App, key, t_j)\} \Sigma_2^*$$

$$\setminus \bigcup_{key_m = key_n} (\Sigma_2 \setminus \{genKey(KGen, key_m, t_j\})^* \{returnKey(KGen, key_n, t_k)\} \Sigma_2^*$$

$$\setminus \bigcup_{key = k(t_j)} (\Sigma_2 \setminus \{genKey(KGen, key, t_j\})^*$$

– As above, we focus on the confidentiality of the key returned to the application, hence μ_2 maps $returnKey(KGen, key, t_j)$ onto $(returnKey(KGen), key)$ and all other actions onto the empty word.

Also in this system model, it is easy to see that the returned key is parameter confidential for Eve regarding μ_2 and *(L,M)-completeness* regarding the same L and set of possible keys M.
Following Definition 3 we can now construct the composed system S_0 with $\Sigma_{IF} = \bigcup_{t\in T, key\in\mathcal{K}}\{callGenKey(App,t), returnKey(KGen,key,t)\}$:

- $\mathbb{P}_0 = \{App, KGen, Eve\}$
- $\Sigma_0 = \bigcup_{t\in T,key\in\mathcal{K}}\{\ boot(t), callGenKey(App,t),$
  $genKey(KGen,key,t), returnKey(KGen,key,t)\}$
- $\lambda^0_{Eve}(boot(t)) = (boot(t),\varepsilon), \forall a \in \Sigma_0 \setminus \bigcup_{t\in T}\{boot(t)\} : \lambda^0_{Eve}(a) = (\varepsilon,\varepsilon)$
- $W^0_{Eve} = \Sigma_0^* \setminus \bigcup\limits_{t_j - t_i = \delta_1} (\Sigma_0 \setminus \{boot(t_i)\})^* \{callGenKey(App,t_j)\}\Sigma_0^*$
  $\setminus \bigcup\limits_{t_k - t_j = \delta_2} (\Sigma_0 \setminus \{callGenKey(App,t_j)\})^*$
  $\{genKey(App,key,t_k)\}\Sigma_0^*$

  $\setminus \bigcup\limits_{key_m = key_n} (\Sigma_0 \setminus \{genKey(KGen,key_m,t_k\})^*$
  $\{returnKey(KGen,key_n,t_l)\}\Sigma_0^*$

  $\setminus \bigcup\limits_{key = k(t_j)} (\Sigma_0 \setminus \{genKey(KGen,key,t_k\})^*$

The question that now needs to be answered is whether or not confidentiality is preserved in this system composition. In the following section, we will introduce theorems that can be used to answer this question.

5 Investigating the Composition of Confidentiality

In this section we provide sufficient conditions under which a composition of two systems preserves the confidentiality properties of each of its components. We start with a very generic approach that is most broadly applicable – however depends on concrete inquiry regarding the satisfaction of the sufficient conditions. Then we provide two more specialized conditions that are less broadly applicable but easier testable.

For each of these cases we first provide a verbal explanation of the concept and then its formal representation. Readers not interested in these formalizations may skip the latter parts. The formalizations all refer to the representation of composition as described in the previous section. An application to the example scenario will be given in Section 6.

For the proofs in this Section we utilize the following lemmata and considerations: The first lemma provides a relation between the local view in the composed system based on the local views from each of the component systems within the integration. This directly reflects the construction rules from Definition 3:

Lemma 1. $(\lambda^0_P)^{-1}(\lambda^0_P(\omega)) = h_1^{-1}((\lambda^1_P)^{-1}(\lambda^1_P(h_1(\omega))))\cap$
$h_2^{-1}((\lambda^2_P)^{-1}(\lambda^2_P(h_2(\omega))))$

Proof. $(\lambda^0_P)^{-1}(\lambda^0_P(\omega)) = (\lambda^0_P)^{-1}((\lambda^{1'}_P(\omega), \lambda^{2'}_P(\omega))$
$= (\lambda^{1'}_P)^{-1}(\lambda^{1'}_P(\omega)) \cap (\lambda^{2'}_P)^{-1}(\lambda^{2'}_P(\omega))$
$= h_1^{-1}((\lambda^1_P)^{-1}(\lambda^1_P(h_1(\omega)))) \cap h_2^{-1}((\lambda^2_P)^{-1}(\lambda^2_P(h_2(\omega))))$

Given a composition we need to find the traces of actions in Component 1 that correspond to those traces in Component 2 – and vice versa. The construction of these relations can be performed via the interface system S^i as well as via the composed system S^c as expressed by the following lemma:

Lemma 2. *Given a system composition as in Definition 3,* $h_1 \circ h_2^{-1} = (h_1^{IF})^{-1} \circ h_2^{IF}$.

Proof. For $x \in \Sigma_2^*$ always holds $h_2^{IF}(x) = h_1(x)$. For $\Diamond$ denoting the shuffle product, $h_1(h_2^{-1}(x)) = h_1(x \Diamond (\Sigma_1 \setminus \Sigma_2)^*) = h_1(x \Diamond (\Sigma_1 \setminus \Sigma_{IF})^*) = h_1(x) \Diamond (\Sigma_1 \setminus \Sigma_{IF})^* = h_2^{IF}(x) \Diamond (\Sigma_1 \setminus \Sigma_{IF})^* = (h_1^{IF})^{-1}(h_2^{IF}(x))$.

For arbitrary sets X and Y and $A, C \subseteq X$, $B, D \subseteq Y$ and a mapping $f : X \longrightarrow Y$ we always have the equality $f^{-1}(B) \cap f^{-1}(D) = f^{-1}(B \cap D)$, but only the inclusion $f(A \cap C) \subseteq f(A) \cap f(C)$. However, for particular intersections we have equality:

Lemma 3. *Let* X, Y *be arbitrary sets,* $f : X \longrightarrow Y$ *a mapping, and* $A \subseteq X, B \subseteq Y$. *Then* $f(A \cap f^{-1}(B)) = f(A) \cap B$.

For the proof of this lemma we refer the reader to [9].

5.1 General Conditions for Confidentiality Composition

The definition of confidentiality in SeMF relies on the extraction and testing of those actions and data that are identified as being confidential. This extraction is applied to every state that the system may take and bases on what an attacker has observed up to this point and what she can deduce from these observations through her initial knowledge.

When two systems that both provide confidentiality are composed into a new system (w.r.t. to some common interface), the conclusion about some data that an attacker may derive at any given state in the composed system is the combination of conclusions she has derived with regards to each of the components. If this combination results in what the attacker is allowed to know in the system composition, then obviously confidentiality is satisfied in the composition.

Within the semantics for confidentiality of SeMF this combination of conclusions about the sequences that may have happened in the individual systems and the value of data used in these sequences is represented as the intersection of these sets – i.e. the smaller a set becomes the more conclusions an attacker can draw, because she considers less values as possible candidates for the confidential data.

It should be noted though that these considerations have to be executed for every state – i.e. every possible sequence of actions – that the system may take. Further, they require a level of detail that would allow for the direct assessment of confidentiality of the composed system instead. However, while these conditions are of less practical relevance, they form the basis for the more restricted conditions presented in the subsequent sections. Formally this approach can be expressed as follows:

Definition 4 *Given a composition as defined in Definition 3, we call* h_1 *confidentiality composable with* h_2 *for R with respect to* μ_0, μ_1 *and* μ_2, *if for all* $\omega \in B_0$ *holds:*

$$\mu_0[\Lambda_{R0}(\omega, W_R^0)] = \mu_1[\Lambda_{R1}(h_1(\omega), W_R^1)] \cap \mu_2[\Lambda_{R2}(h_2(\omega), W_R^2)]$$

Theorem 1 *Given a confidentiality composable composition as defined in Definition 4, if S_1 and S_2 both are parameter confidential for agent R with respect to some μ_1 and μ_2 with $\mu_1 \circ h_1 = \mu_2 \circ h_2$, then S_0 is parameter confidential for R with respect to $\mu_0 := \mu_1 \circ h_1 = \mu_2 \circ h_2$ and $L_0 := L_1 = L_2, M_0 := M_1 = M_2$.*

Proof. S_1, S_2 parameter confidential, $h_1(B_0) \subseteq B_1$, $h_2(B_0) \subseteq B_2$ implies $\forall \omega \in B_0$:
$\mu_1[\Lambda_{R1}(h_1(\omega), W_R^1)] \supseteq p_t^{-1}(p_t(\mu_1[\Lambda_{R1}(h_1(\omega), W_R^1)])) \cap K$ and
$\mu_2[\Lambda_{R2}(h_2(\omega), W_R^2)] \supseteq p_t^{-1}(p_t(\mu_2[\Lambda_{R2}(h_2(\omega), W_R^2)])) \cap K$.
Taking the intersection of these equations leads to
$\mu_1[\Lambda_{R1}(h_1(\omega), W_R^1)] \cap \mu_2[\Lambda_{R2}(h_2(\omega), W_R^2)]$
$\supseteq p_t^{-1}(p_t(\mu_1[\Lambda_{R1}(h_1(\omega), W_R^1)])) \cap p_t^{-1}(p_t(\mu_2[\Lambda_{R2}(h_2(\omega), W_R^2)])) \cap K$
$= p_t^{-1}[p_t(\mu_1[\Lambda_{R1}(h_1(\omega), W_R^1)]) \cap p_t(\mu_2[\Lambda_{R2}(h_2(\omega), W_R^2)]) \cap K]$
$\supseteq p_t^{-1}(p_t(\mu_1[\Lambda_{R1}(h_1(\omega), W_R^1)] \cap \mu_2[\Lambda_{R2}(h_2(\omega), W_R^2)])) \cap K$.
By assumption of h_1 and h_2 being confidentiality composable it follows that
$\mu_0[\Lambda_{R0}(\omega, W_R^0)] = p_t^{-1}(p_t(\mu_0[\Lambda_{R0}(\omega, W_R^0)])) \cap K$.

5.2 Independantly Testable Conditions for Confidentiality Composition

Testing for the confidentiality of data by analysing data values considered possible by the attacker, as presented in the previous approach, is performed on the same level of detail as the direct assessment of confidentiality. In the approach presented in this section, we instead perform an assessment of the usage of the interface by the composed components regarding observations and knowledge that can be gained by an attacker.

Following this approach it is possible for two component designers to agree about the information regarding the components' interface that an attacker may get and thereby allows for a more distributed development of each of the components.

For a given state (i.e. sequence of actions) in the composed system, the conclusions regarding the interface behaviour that an attacker can draw from her observations and initial knowledge from each of the components must be equal. Consequently, during the design of the interface the component designers must agree on the interface behaviour that shall be considered possible by the attacker when observing the behaviour of the individual components.

The interaction of designers can be further decoupled by overestimating the set of possible states: Instead of considering all possible states / sequences of actions of the composed system, the designers may only define the set of possible sequences of actions at the interface (interface behaviour). This set can then be associated with the sequences considered possible by the attacker in each of the components, which leads to an agreement over the attacker's deductive capabilities.

The component designers can then independently assess if their component fulfils this requirement (equality of interface behaviour concluded from the individual components) by focussing on all sequences of actions that their component can take that will result in one of the agreed interface behaviour sequences.

Definition 5 *A composition following Definition 3 is called confidentiality preserving if the following assumption holds for all $P \in \mathbb{P}_0, \omega \in B_0$:*

$$a)\ h_1^{IF}((\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1) = h_2^{IF}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega))) \cap W_P^2)$$

Alternatively, for all $P \in \mathbb{P}_0, \omega \in B^{IF}$, $\omega_1 \in (h_1^{IF})^{-1}(\omega), \omega_2 \in (h_2^{IF})^{-1}(\omega)$*:*

$$b)\ h_1^{IF}((\lambda_P^1)^{-1}(\lambda_P^1(\omega_1)) \cap W_P^1) = h_2^{IF}((\lambda_P^2)^{-1}(\lambda_P^2(\omega_2)) \cap W_P^2)$$

which implies condition a) by overestimation of possible component state combinations.

Theorem 2 *Given a confidentiality preserving composition according to Definition 5 and given that system* S_1 *has a confidentiality property w.r.t. some* μ_1 *and* K *then* S_0 *has the confidentiality property regarding* $\mu_0 = \mu_1 \circ h_1$ *and the same* K*.*

Proof. $\mu_0[\lambda_0^{-1}(\lambda_1(\omega)) \cap W_0] = \mu_1(h_1[\lambda_0^{-1}(\lambda_1(\omega)) \cap W_0])$
$= \mu_1(h_1[\lambda_0^{-1}(\lambda_1(\omega)) \cap h_1^{-1}(W_P^1) \cap h_2^{-1}(W_P^2)])$
Using Lemma 3 leads to equality with
$\mu_1[W_P^1 \cap h_1(\lambda_0^{-1}(\lambda_1(\omega)) \cap h_2^{-1}(W_P^2))]$
$= \mu_1 W_P^1 \cap [h_1(h_1^{-1}((\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega)))) \cap h_2^{-1}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega)))) \cap h_2^{-1}(W_P^2))]$
Again applying Lemma 3 implies equality with
$\mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1 \cap h_1(h_2^{-1}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega)))) \cap h_2^{-1}(W_P^2))]$
$= \mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1 \cap h_1(h_2^{-1}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega)))) \cap W_P^2)]$
Applying Lemma 2 leads to equality with
$= \mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1 \cap h_1^{IF^{-1}}(h_2^{IF}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega)))) \cap W_P^2)]$
which by Assumption equals
$\mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1 \cap h_1^{IF^{-1}}(h_1^{IF}((\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega)))) \cap W_P^1)]$
which is finally equal to
$\mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1]$ which concludes our proof.

5.3 Design of Generally Composable Component Interfaces

This final approach for composition targets the design of interfaces between components. The goal is to design the interface between two components in such a way that no additional considerations have to be made when composing confidentiality properties.

This is for example possible if an interface handles only the single transfer of confidential data. Obviously, if this data is handled in a confidential way by both components, there cannot be any side effects within the interface that may destroy the confidentiality property. This is expressed by testing that for any two combinations of sequences of actions within the interface, the extraction of confidential data from their combination will equal those candidates that result from the combinations of candidates derived independently from each of the sequences.

Most notably in this approach, it is not necessary to assess the capabilities (in terms of local view and initial knowledge) of a possible attacker. The design of the interface will make it impossible for any attacker to gain advantage by the composition of the components as long as they each provide confidentiality of the data. Formally, this is expressed as:

Definition 6 *A composition following Definition 3 has a generally composable interface with respect to some* μ_{IF} *if*

$$\forall A \subseteq h_1^{IF}(W_P^1), B \subseteq h_2^{IF}(W_P^2) : \mu_{IF}(A \cap B) = \mu_{IF}(A) \cap \mu_{IF}(B)$$

Trivially, if μ_{IF} is an isomorphism, the above property is implied.

Theorem 3 *Given a generally composable interface composition as defined in Definition 6, if S_1 and S_2 both are parameter confidential for agent R with respect to some μ_1 and μ_2 , then S_0 is parameter confidential for R with respect to $\mu_0 = \mu_1 \circ h_1 = \mu_2 \circ h_2 = \mu_{IF} \circ h'_1 \circ h_1 = \mu_{IF} \circ h'_2 \circ h_2$.*

Proof. $\mu_0[\lambda_0^{-1}(\lambda_1(\omega)) \cap W_0] = \mu_1(h_1[\lambda_0^{-1}(\lambda_1(\omega)) \cap W_0])$
$= \mu_1(h_1[\lambda_0^{-1}(\lambda_1(\omega)) \cap h_1^{-1}(W_P^1) \cap h_2^{-1}(W_P^2)])$.
Using Lemma 3 leads to equality to
$\mu_1[W_P^1 \cap h_1(\lambda_0^{-1}(\lambda_1(\omega)) \cap h_2^{-1}(W_P^2))]$
$= \mu_1 W_P^1 \cap [h_1(h_1^{-1}((\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega)))) \cap h_2^{-1}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega)))) \cap h_2^{-1}(W_P^2))]$
By Lemma 3 this is equal to
$\mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1 \cap h_1(h_2^{-1}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega)))) \cap h_2^{-1}(W_P^2))]$
$= \mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1 \cap h_1(h_2^{-1}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega))) \cap W_P^2))]$
By Lemma 2 this is equal to
$\mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1 \cap h_1^{IF^{-1}}(h_2^{IF}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega))) \cap W_P^2))]$
As $\mu_1 = \mu_{IF} \circ h_1^{IF}$ and using Lemma 3 leads to equality with
$\mu_{IF}[h_1^{IF}((\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1) \cap h_2^{IF}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega))) \cap W_P^2)]$.
By assumption, this equals
$\mu_{IF}[h_1^{IF}((\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1)] \cap \mu_{IF}[h_2^{IF}((\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega))) \cap W_P^2)]$
$= \mu_1[(\lambda_P^1)^{-1}(\lambda_P^1(h_1(\omega))) \cap W_P^1] \cap \mu_2[(\lambda_P^2)^{-1}(\lambda_P^2(h_2(\omega))) \cap W_P^2]$
which satisfies Definition 4.

6 Revisiting the Scenarios

In this section, we revisit the scenario composition introduced in Section 4.2 and demonstrate where and how this composition fails with regards to the formal considerations presented in Section 5. From the description in Section 2 it is already known that the example scenario does not preserve confidentiality during composition. In this section we demonstrate how our sufficient conditions, if not met, give hints regarding the possible reasons of confidentiality being violated in the composition, and how the components can be changed in order to preserve confidentiality.

For the following illustrations, we do not require the point in time at which a key is returned for assessing the confidentiality of the key; hence we define $\Sigma_t := \{returnKey(KGen)\}$. For the ease of reading we further simplify the system by restricting it to one single run; i.e. $\forall a \in \Sigma, \omega \in B : card(a, alph(\omega)) = 1$. This results in a considerable reduction of complexity but does not affect the applicability of our methods. Analogous results can be obtained for the full system behaviour.

6.1 General Conditions for Confidentiality Composition

Following the system definitions in Section 4.2 we investigate the preservation of confidentiality in the example composition. We demonstrate that Theorem 1 is not applicable and show how this fact can be used to identify the side effects that violate the confidentiality in the composed system. We use the following sequence of actions:

$$\omega_0 = boot(t_1)\ callGenKey(App, t_2)\ genKey(KGen, key_0, t_3)$$
$$returnKey(KGen, key_0, t_4) \text{ with } t_2 = t_1 + \delta_1 \text{ and } t_3 = t_2 + \delta_2$$

We start by assessing the left hand side of the equation of Definition 4, followed by the two sets for the right hand side.

Given ω_0, we can assess the sequences that Eve considers possible in S_0 (with $pre(\omega)$ denoting the set of prefixes of ω):

$$\Lambda^0_{Eve}(\omega_0, W^0_{Eve}) = \\ pre[\bigcup_{t_x \in \mathcal{T}} \{boot(t_1)\ (callGenKey(App, t_1 + \delta_1)\ genKey(KGen, key_0, t_1 + \delta_1 + \delta_2)\ returnKey(KGen, key_0, t_x)\}] \\ \setminus \{\varepsilon\}$$

Since Eve knows δ_1 and δ_2, and since the key is completely determined by its time of generation, she only considers one value possible for the returned key

$$\mu_0[\Lambda^0_{Eve}(\omega_0, W^0_{Eve})] = \{(returnKey(KGen), key_0)\} \text{ with } key_0 = k(t_1 + \delta_1 + \delta_2)$$

Regarding the conception of Eve with respect to each of the component systems, we again assess all those sequences that Eve considers possible for the respective images of ω_0 in these systems:

$$\Lambda^1_{Eve}(h_1(\omega_0), W^1_{Eve}) = pre[\bigcup_{t_x \in \mathcal{T}, key_i \in \mathcal{K}} \{boot(t_1)\ callGenKey(App, t_1 + \delta_1)\ returnKey(KGen, key_i, t_x)] \setminus \{\varepsilon\}$$

$$\Lambda^2_{Eve}(h_2(\omega_0), W^2_{Eve}) = pre[\bigcup_{t_x, t_y \in \mathcal{T}} \{callGenKey(App, t_x)\ genKey(KGen, key_j, t_x + \delta_2)\ returnKey(KGen, key_j, t_y)\}] \\ \text{with } key_j = k(t_x + \delta_2)$$

This leads to the following sets of values that Eve considers as candidates for the confidential data (as t_x originates from all of $\mathcal{T}$, every $key_i \in \mathcal{K}$ is possible):

$$\mu^1[\Lambda^1_{Eve}(h_1(\omega_0), W^1_{Eve})] = \bigcup_{key_i \in \mathcal{K}} \{(returnKey(KGen), key_i)\}$$

$$\mu^2[\Lambda^2_{Eve}(h_2(\omega_0), W^2_{Eve})] = \bigcup_{key_j \in \mathcal{K}} \{(returnKey(KGen), key_j)\} \cup \{\varepsilon\}$$

Coming back to Definition 4 we can see that the values considered possible by Eve in the composition do not equal the combined (i.e. intersected) knowledge from each of the component systems:

$$\{(returnKey(KGen), key_0\} \neq \big(\bigcup_{key_i \in \mathcal{K}} \{(returnKey(KGen), key_i)\}\big) \\ \cap \big(\bigcup_{key_j \in \mathcal{K}} \{(returnKey(KGen), key_j)\} \cup \{\varepsilon\}\big)$$

implies $\mu_0[\Lambda^0_{Eve}(\omega_0, W^0_{Eve})] \neq \mu^1[\Lambda^1_{Eve}(h_1(\omega_0), W^1_{Eve})] \cap \mu^2[\Lambda^2_{Eve}(h_2(\omega_0), W^2_{Eve})]$

It can be seen however, that if t_1 or δ_1 were unknown to Eve, the confidentiality would be preserved. This relates to the use case as Desktop Application where an attacker does not know at which point in time a user initiates a key generation. It can further be seen that if key was not derived from these values but for example from a non-pseudo random number generator, Eve would also not be able to derive the key's value in the composition.

6.2 Independantly Testable Conditions for Confidentiality Composition

Similarly, Definition 5 can be used to illustrate that the condition of Theorem 2 sufficient for preserving confidentiality does not hold. Using the same ω_0 as in the previous section results in the same sets $\Lambda^1_{Eve}(h_1(\ \omega_0), W^1_{Eve})$ and $\Lambda^2_{Eve}(h_2(\omega_0), W^2_{Eve})$. We now investigate the projections of these sets into the interface system in order to compare the interface expectations of both components.

$$
\begin{aligned}
h_1^{IF}[\Lambda^1_{Eve}(h_1(\omega_0),W^1_{Eve})] = pre[& \bigcup_{t_x \in \mathcal{T}, key_i \in \mathcal{K}} \{ \\
& callGenKey(App, t_1 + \delta_1)\ returnKey(KGen, key_i, t_x)] \setminus \{\varepsilon\} \\
h_2^{IF}[\Lambda^2_{Eve}(h_2(\omega_0),W^2_{Eve})] = pre[& \bigcup_{t_y, t_z \in \mathcal{T}} \{ \\
& callGenKey(App, t_y)\ returnKey(KGen, key_i, t_z)\}] \\
& \text{with } key_i = k(t_y + \delta_2)
\end{aligned}
$$

As we can see, these sets are not equal. The dependence of key_i on the point in time of $callGenKey$ being performed is not expected by the App component, which hints to the confidentiality preservation error.

In order to avoid such a situation, the developers of the components could have agreed a priory to a common assumed interface behaviour when they agreed on the interface design. Following option b) of Definition 5 this could have been

$$
B^{IF} = pre[\bigcup_{t_x < t_y - \delta \in \mathcal{T}, key_i \in \mathcal{K}} callGenKey(App, t_x)\ returnKey(KGen, key_i, t_y)]
$$

In this case the developer of the key generator would have needed to alter his/her implementation to reflect the functional independence of t_x and key_i, leading to a confidentiality preserving composition.

6.3 Design of Generally Composable Component Interfaces

Finally, we demonstrate that our example scenario does not satisfy the sufficient condition specified in Definition 6 and show how in particular scenarios the system specification can be corrected in order for the condition to hold and thus confidentiality to hold as well in the composition. We choose the following two sequences of actions from the respective sets:

- $h_1^{IF}(W_{Eve}^1) \ni A = \{callGenKey(App, t_1)\ returnKey(KGen, key_A, t_y)\}$ with $key_A \in \mathcal{K}$ (key_A can be chosen independently of t_1).
- $h_2^{IF}(W_{Eve}^2) \ni B = \{callGenKey(App, t_2)\ returnKey(KGen, key_B, t_y)\}$ with $key_B = k(t_2 + \delta_2)$ according to S_2.

Obviously, as for $t_1 \neq t_2$ A and B are distinct sets, $\mu_{IF}(A \cap B) = \emptyset$. However, for $key_A = k(t_2 + \delta_2) = key_B$, it follows $\mu(A) = \mu(B) = \{(returnKey(KGen), key_A)\} = \mu_{IF}(A) \cap \mu_{IF}(B)$.

In order to construct a system that fulfills the condition for a generally composable interface, S_{IF} must be designed in such a way that μ_{IF} is an isomorphism. This is the case e.g. if the interface only consists of a stream of generated keys that are handed over from the key generator to the application with $\Sigma_{IF} = \{provideKey(KGen, key_i)\}$. As there exists no functional relation from App to $KeyGen$ there cannot be side-effects that destroy the confidentiality property on the key generator's side during composition.

7 Related Work

The model based composition of systems is a field of growing research activity in the last decade. Tout et al. [14] have developed a methodology for the composition of web services with security. They use the Business Process Execution Language (BPEL) for the specification of web services composition and expand it in order to specify the security properties independently from the business logic based on policy languages using a UML Profile for specifying the required security properties. Their approach focusses on how to specify security requirements of web service compositions and does not address verification of security properties in such compositions. Sun et al. propose in [15] a service decomposition-based approach for service composition in which the utility of a composite service can be computed from the utilities of component services, and the constraints of components services can be derived from the constraints of the composite service. Their approach manages the selection of each component service, leading to more scalability and more flexibility for service composition in a dynamic environment. However, this approach focusses on maximizing the utility of the composition and does not address security properties. A method for composing a system from service components with anonymous dependencies is presented by Sora et al. in [16]. They specify component descriptions by means of semantic-unaware properties, an application-domain independent formalism for describing the client-specific configuration requests in terms of desired properties, and propose a composition algorithm. Using a different approach, Lei Zhang and Jun Wu [17] analyse the relationship between trustworthiness attributes and propose models of these attributes and their relationship. They use a Trustworthy Software Composition Architecture (TSCA) software as evaluation method.

Rossi presents in [18] a logic-based technique for verifying both security and correctness properties of multilevel service compositions. Service compositions are specified in terms of behavioural contracts which provide abstract descriptions of system behaviours by means of terms of a process algebra. Multi-party service compositions are modelled as the parallel composition of such contracts. Modal mu-calculus formulae are used to characterize non-interference and compliance (i.e. deadlock and livelock free) properties. The well-known concepts of non-interference or information flow control address

confidentiality with respect to actions. In the above approach, these concepts are used to specify that public synchronizations (i.e. actions concerned with the communication between services) are unchanged as confidential communications are varied. Hence it is not clear how this approach can be extended to cover cases in which satisfaction of confidentiality depends solely on whether specific parameters of an action are visible.

Universal Composability is another prominent branch of research addressing the composition of cryptographic protocols while preserving certain security properties (see for example [19–21]). A common paradigm in this area of research is that a protocol that "securely realizes" its task is equivalent to running an idealized computational process (also called "ideal functionality") where security is guaranteed. A main disadvantage of the Universal Composability approach seems to be that for every property that shall be proven, a new ideal process has to be constructed whose interactions with the parties result in providing this property.

Pino et al. present in [22] an approach for constructing secure service compositions, making use of composition patterns and security rules. They prove integrity and confidentiality of service compositions based on specific security properties provided by the individual components of such a composition. While the proofs are based on the same formal framework as the one presented in this paper, their approach uses an intermediate orchestration component. We in contrast focus on the direct composition of any type of components, deriving security proofs from specific conditions concerning the component interfaces.

8 Conclusions & Future Work

In this paper we presented the formalization of the composition of two systems that allows to formally reason about the preservation of confidentiality properties. The central idea is to view each of the systems as an abstraction of their composition, and to describe each aspect of the composition (e.g. its behaviour, agents' local views and initial knowledge) in terms of these abstractions. We then introduced conditions that allow to prove that a specific confidentiality property holds for the composition if it holds for the individual components. Using the composition of an application with a key generation module as scenario, we then demonstrated that the fact that these conditions do not hold reveals side effects with non-trivial implications regarding confidentiality. In particular, we presented a general sufficient condition for preservation of confidentiality that is of more theoretical interest, and derived two more specific conditions that are applicable in distributed system engineering and point to particular aspects of the two components that need to be taken into consideration by the developers. The first concerns additional agreements on interface level between component developers that can be independently tested for each component, the second provides sufficient conditions regarding the interface itself that rules out side effects during composition and thereby guarantees the preservation during composition of any two components that implement these interfaces.

Currently we are working on other types of conditions sufficient for proving confidentiality of a system. Finding relations of these conditions to the ones presented in this paper may broaden their scope of application. Future work includes the application of

the foundations layed out in this paper to general software engineering by projecting the semantic knowledge onto rules and guidelines for composition of software components.

References

1. Heise News: Neuer Personalausweis: AusweisApp mit Lücken. heise.de (2010)
2. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your Ps and Qs: Detection of widespread weak keys in network devices. In: Proceedings of the 21st USENIX Security Symposium. (August 2012)
3. Anderson, R.: Security Engineering: A guide to building dependable distributed systems. Wiley (2010)
4. SERENITY Consortium: Serenity (2006) http://www.serenity-project.org.
5. TERESA Consortium: Trusted computing Engineering for Resource constrained Embedded Systems Applications (2009) http://www.teresa-project.org/.
6. SecFutur Consortium: SecFutur (2010) http://www.secfutur.eu/.
7. 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
8. Gürgens, S., Ochsenschläger, P., Rudolph, C.: Parameter confidentiality. In: Informatik 2003 - Teiltagung Sicherheit, Gesellschaft für Informatik (2003)
9. Gürgens, S., Ochsenschläger, P., Rudolph, C.: Abstractions preserving parameter confidentiality. In: European Symposium On Research in Computer Security (ESORICS 2005). (2005) 418–437
10. Kerrisk, M.: LCE: Don't play dice with random numbers. LWN.net (2012)
11. Corbet, J.: Random numbers for embedded devices. LWN.net (2012)
12. Fuchs, A., Gürgens, S., Rudolph, C.: A Formal Notion of Trust – Enabling Reasoning about Security Properties. In: Proceedings of Fourth IFIP WG 11.1 International Conference on Trust Management. (2010)
13. Fuchs, A., Gürgens, S., Rudolph, C.: Formal Notions of Trust and Confidentiality - Enabling Reasoning about System Security. Journal of Information Processing **19** (2011) 274–291
14. Tout, H.e.a.: Towards a bpel model-driven approach for web services security. In: International Conference on Privacy, Security and Trust (PST'12). (2012)
15. Sun, SX.; Zhao, J.: A decomposition-based approach for service composition with global qos guarantees. In: Journal of Information Sciences. (2012)
16. Sora, L., et al.: Automatic composition of systems from cponents with anonymous dependencies specified by semantic-unaware properties. In: Technology of object-oriented languages, systems and architecture. (2003)
17. Zhang, L., Wu, J.: Research on trustworthy software composition architecture. In: International Conference on Consumer Electronics, Communications and Networks. (2012)
18. Rossi, S.: Model checking adaptive multilevel service compositions. In: International Workshop of Formal Aspects of Component Software. (2010)
19. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology **13**(1) (2000) 143–202
20. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: focs, Published by the IEEE Computer Society (2001) 136
21. Canetti, R., Herzog, J.: Universally composable symbolic analysis of mutual authentication and key-exchange protocols. Theory of Cryptography (2006) 380–403
22. Pino, L., Spanoudakis, G.: Constructing secure service compositions with patterns. In: Services (SERVICES), 2012 IEEE Eighth World Congress on, IEEE (2012) 184–191