Parameter – Confidentiality[*]

Sigrid Gürgens, Peter Ochsenschläger, Carsten Rudolph
Fraunhofer – Institute Secure Telecooperation, Germany
{guergens,ochsenschlaeger,rudolphc}@sit.fraunhofer.de

Abstract: Confidentiality of certain parameters is an essential security requirement for many security sensitive applications. In terms of formal language theory we introduce the notion of parameter-confidentiality relative to an agent's knowledge about the system. By considering publicly known dependencies of parameter values, exact specifications of the required confidentiality properties are possible. The new notion complements previous concepts of non-interference, secrecy and indistinguishability.

1 Introduction

In many security sensitive systems *confidentiality* plays a central role: keys for decryption should be confidential, the occurrence of certain actions as for instance access to a data base perhaps has to be confidential, and different prices offered to different customers may have to be confidential as well. A closer look at these examples shows that the term *confidentiality* is used with different meanings.

On the other hand, formal security models as well as security analysis and design of security sensitive systems need precise definitions of the security goals. So the variety of meanings of the term *confidentiality* shows the necessity to formalize its different aspects, which we will illustrate now using the three above examples.

Confidentiality of a key requires that the key cannot be guessed, because a correct guess of a key used for decryption can be verified by an attacker if the plaintext is either known or contains redundancy, which drastically violates the desired security property. However, in practice non-guessability is not realizable. Therefore, the keyspace must be chosen in a way that a key is only correctly guessable with negligible probability. Confidentiality properties of this type are usually formalised using concepts of probability theory or complexity theory [MvOV96].

Confidentiality of occurence of certain actions (like database access) or certain data (like prices offered to a customer) poses a different problem as a correct guess of some action or some data may occur or may even be impossible to prevent. For instance the range of possible prices can be very restricted and therefore a correct guess of the price has a

[*] Part of this work is based on results of a project funded by Siemens AG, and part of this work was developed within the project CASENET funded by the European Commission (IST-2001-32446)

high probability. Consequently, the verification of the guess has to be prevented, i.e. the information necessary to verify the guess must not be known to an attacker.

Confidentiality of actions (as in the data base access example) is typically addressed by the well-known concept of non-interference or information flow control: the occurrence or non-occurrence of certain actions of an agent shall not be deducible for another agent based on what it observes. In the literature there is a variety of formalizations of this concept starting with the work of Goguen and Meseguer [GM82]. Mantel [Man00] gives a good insight into this topic. The subtle differences between these definitions show the spectrum of this kind of confidentiality.

Similar to the data base access example, the price example illustrates a so called possibilistic security property discussing which system behaviour seems to be possible to an agent depending on its observations, and what can it deduce from that. In contrast to non-interference where the goal of such a deduction is the occurrence of certain actions, in the price example the goal are parameter values of certain actions. In this case the occurrence of the actions itself may be known.

A formalization of this third aspect of confidentiality, which we call *parameter-confidentiality*, is the focus of our paper. Motivated by two characteristic electronic commerce examples we present two definitions, which show different aspects of parameter-confidentiality. Similar to the subtle differences in the definitions of non-interference, the differences of our definitions reflect subtle characteristics of the required properties. In particular, our definitions capture correlations of parameter values in different occurrences of actions. Satisfaction of these properties is relative to an agent's view of what has happened in the system and which system behaviour it considers possible according to its knowledge about the system. The definitions are formulated in terms of formal language theory and fit in our design method for security sensitive systems [GO01, Rud01, GOR02].

This method has been successfully applied to authenticity and provability: Security requirements have been formulated on a high level of abstraction. On a lower level they have been realized by so called abstract secure channels modelling cryptographic primitives. Correctness of such a realization has been proven by specific language homomorphisms which transport the security properties.

Parameter-confidentiality is related to the notion of indistinguishability used to define security of public key cryptography [GM84, BDPR98]. This notion is equivalent to the notion of semantic security describing that no information about the plaintext can be deduced from a ciphertext. Semantic secure encryption might be used to implement the transfer of confidential parameter values over insecure communication channels. Our definition however not only captures confidentiality of parameter transmission but also formalizes confidentiality of any kind of information that allows to draw conclusions about the parameter values.

For security analysis of cryptographic protocols in [AG99], indistinguishability of parameter values is introduced in terms of the π*-calculus* and is used to express secrecy of keys, nonces, etc. As it correlates a parameter value to a complete protocol run and as it expresses no assumptions on the knowledge of an attacker it is tailored to key-establishment and authentication protocols and therefore more restrictive than our approach.

2 System behaviour and agent's knowledge about a system

In this section we first give a short summary of the necessary concepts of formal languages to describe system behaviour. Then we describe how an agent P's knowledge about such a system can be formalised.

The behaviour S of a discrete system can be formally described by the set of its possible sequences of actions. Therefore $S \subseteq \Sigma^*$ holds where Σ is the set of all actions of the system, and Σ^* is the set of all finite sequences of elements of Σ, including the empty sequence denoted by ε. This terminology originates from the theory of formal languages [Eil74], where Σ is called the alphabet (not necessarily finite), the elements of Σ are called letters, the elements of Σ^* are referred to as words and the subsets of Σ^* as formal languages. Words can be composed: if u and v are words, then uv is also a word. This operation is called the *concatenation*; especially $\varepsilon u = u\varepsilon = u$. A word u is called a *prefix* of a word v if there is a word x such that $v = ux$. The set of all prefixes of a word u is denoted by pre(u); $\varepsilon \in$ pre(u) holds for every word u.

Formal languages which describe system behaviour have the characteristic that pre$(u) \subseteq S$ holds for every word $u \in S$. Such languages are called *prefix closed*. System behaviour is thus described by prefix closed formal languages.

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 $h^*(\Sigma) \subseteq \Sigma' \cup \{\varepsilon\}$. 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.

Let $\mathbb{P}$ be a set of agents. For each $P \in \mathbb{P}$ we denote by $W_P(S) \subseteq \Sigma^*$ the set of those sequences agent P considers to be possible. $W_P(S)$ formalizes P's knowledge about a system S. If the related system behaviour is obvious, we shortly write W_P.

We assume $S \subseteq W_P$, i.e. every agent considers the system behaviour to be possible. Security properties can now be defined relative to W_P.

After a sequence of actions $\omega \in S$ has happened, every agent can only use its *local view* of ω to determine the sequences of actions it considers to be possible. In order to determine what is the local view of an agent, we first assign every action to exactly one agent. Thus $\Sigma = \dot{\bigcup}_{P \in \mathbb{P}} \Sigma_{/P}$ (where $\Sigma_{/P}$ denotes all actions performed by agent P, and $\dot{\bigcup}$ denotes the disjoint union). The homomorphism $\pi_P : \Sigma^* \to {\Sigma^*}_{/P}$ defined by $\pi_P(x) = x$ if $x \in \Sigma_{/P}$ and $\pi_P(x) = \varepsilon$ if $x \in \Sigma \setminus \Sigma_{/P}$ formalizes the assignment of actions to agents and is called the $projection$ on P.

The projection π_P is the correct representation of P's view of the system if all information about an action $x \in \Sigma_{/P}$ is available for agent P. In automata models, for example, the elements of Σ may contain information about the global system state (e.g. all agents' memory) and may be represented by a triple (*global state*, *transition label*, *global successor state*). However, an agent P generally cannot "see" the complete global state (it cannot see, for example, other agents' memory). Therefore, the projection π_P may be too fine to define the local view of an agent $P \in \mathbb{P}$. Thus, we generally denote the local view of an agent P on Σ by $\lambda_P : \Sigma^* \to \Sigma_P^*$.

For a sequence of actions $\omega \in S$ and agent $P \in \mathbb{P}$, $\lambda_P^{-1}(\lambda_P(\omega)) \subseteq \Sigma^*$ is the set of all sequences that look exactly the same from P's local view after ω has happened. But depending on its knowledge about the system S, underlying security mechanisms and system assumptions, P does not consider all sequences in this set 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.

3 The formal definitions

We want to formalize the following property: An agent R that monitors a sequence of actions ω of a system S cannot distinguish between the possible values of a certain parameter (a certain part of the message, the agent performing the action, etc.) of a specific action or set of actions of the sequence, even if it knows the set of possible parameter values. Consider for example an application consisting of the following actions: a user requests a price for a certain service, the request is received by a service provider and then an offer for this service is sent and received. In this example, one critical parameter might be the price. The service provider might have different rates for different users and these rates can change. We assume the price is supposed to be confidential, i.e. no other agent shall be able to tell which price has been offered. In the remainder of the paper the external agent (the attacker) is denoted R, the user U and the service provider SP. The actions in the system are *send-price-request(U,SP), rec-price-request(SP,U), send-offer(SP,U,price)* and *rec-offer(U,SP,price)*. The first parameter denotes the agent executing the particular action. An additional observe action *obs* is defined for R which enables R to learn some information about the previous action. Therefore, *obs* has only one parameter representing the previous action. However, R may not be able to learn all parameters of the previous action. What R learns from *obs* actions is specified in the set W_R which corresponds to all sequences of actions R considers to be possible. We say R *monitors* a sequence ω if after each action of U or SP an *obs* action is executed. R can only "see" actions performed by itself. Consequently, in the example, R's local view $\lambda_R(\omega)$ is defined as $\lambda_R(\omega) := \pi_R(\omega)$ and contains only $observe$ actions.

In the sequences of actions that R considers possible after having monitored ω, only the actions where a price is sent and received are of interest. Thus we disregard all other actions, i.e. we map them with a suitably chosen homomorphism μ onto the empty word. From those actions not mapped onto ε, μ extracts the confidential parameter that occurs in the action. Generally not only the parameter itself but also the "type" of its occurrence has to be considered. This type can be, for example, that a certain user U has received an offer. The parameter associated with this type is the price included in the offer. By considering only the type, actions from Σ are divided into classes the elements of which can be distinguished essentially by the parameter values. Each of these classes is represented by one type.

Hence $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ is a set of sequences of actions that consist of the types of those actions that are of interest with respect to parameter confidentiality, paired with the respective parameter values being possible from R's local view.

If Σ_t denotes the set of types of the parameter occurrences and if M denotes the set of parameter values then $\mu_{\Sigma_t,M} : \Sigma^* \rightarrow (\Sigma_t \times M)^*$ is a homomorphism. For simplicity we write μ if the related parameter set and the types are obvious. Such a homomorphism μ can be defined as follows:

$$\begin{array}{lcl}
\mu(\textit{send-offer}(SP, U, \textit{price})) & = & (Send_{SP}, \textit{price}) \\
\mu(\textit{rec-offer}(U, SP, \textit{price}) & = & (Rec_U, \textit{price}) \\
\mu(\textit{send-price-request}(U, SP)) & = & \mu(\textit{rec-price-request}(SP, U)) = \mu(obs(\ldots)) = \varepsilon
\end{array}$$

In order to explain our formalism, we use the price offer system S. Our aim is now to formalize that $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ "contains all possible parameter values".

3.1 (L, M)–Completeness

For the following, we assume that R monitors all sequences of actions. Let us consider as an example the following sequence of actions:

$$\begin{array}{lcl}
\omega & = & \textit{send-price-request(U,SP) obs(send-price-request(U,SP))} \\
& & \textit{rec-price-request(SP,U) obs(rec-price-request(SP,U))} \\
& & \textit{send-offer(SP,U,price}_1\textit{) obs(send-offer(SP,U,price}_1\textit{))} \\
& & \textit{rec-offer(U,SP,price}_1\textit{) obs(rec-offer(U,SP,price}_1\textit{))} \\
& & \textit{send-price-request(U,SP) obs(send-price-request(U,SP))} \\
& & \textit{rec-price-request(SP,U) obs(rec-price-request(SP,U))} \\
& & \textit{send-offer(SP,U,price}_2\textit{) obs(send-offer(SP,U,price}_2\textit{))}
\end{array}$$

Let us further assume that for $R \neq U, SP$ it shall be confidential which price was sent and received, respectively. Let $\{price_1, price_2\}$ be the set of possible prices, and Σ the set of resulting possible actions. As described above, R's local view of this sequence is the following:

$$\begin{array}{lcl}
\lambda_R(\omega) & = & \textit{obs(send-price-request(U,SP)) obs(rec-price-request(SP,U))} \\
& & \textit{obs(send-offer(SP,U,price}_1\textit{)) obs(rec-offer(U,SP,price}_1\textit{))} \\
& & \textit{obs(send-price-request(U,SP)) obs(rec-price-request(SP,U))} \\
& & \textit{obs(send-offer(SP,U,price}_2\textit{))}
\end{array}$$

Now if R does not know which of the possible two parameters was sent, but does know that the same parameter that was sent was also received, R considers four different types of actions possible: two in which SP sends and U receives twice the same parameter (either $price_1$ or $price_2$), one in which first $price_1$ is sent and received and then $price_2$, and one in which the parameters are sent and received in reverse order.

The function μ now maps these sequences of actions onto sequences with types for the send and receive actions, each one being paired with the respective parameter. All other actions, including the $observe$ actions, are mapped onto ε. This results in

$$\begin{aligned}\mu(\lambda_R^{-1}(\lambda_R(\omega))\cap W_R) \quad = \quad & \{(Send_{SP},price_1)(Rec_U,price_1)(Send_{SP},price_1),\\ & (Send_{SP},price_1)(Rec_U,price_1)(Send_{SP},price_2),\\ & (Send_{SP},price_2)(Rec_U,price_2)(Send_{SP},price_1),\\ & (Send_{SP},price_2)(Rec_U,price_2)(Send_{SP},price_2)\}\end{aligned}$$

If we want to describe a situation where R does not know any correlation between the parameter of a send and the respective receive action, the μ-image of the sequence of actions monitored by R contains eight different sequences of pairs $(type, parameter)$ with no order on the parameters $price_1$ and $price_2$:

$$\begin{aligned}\mu(\lambda_R^{-1}(\lambda_R(\omega))\cap W_R) \quad = \quad & \{(Send_{SP},price_1)(Rec_U,price_1)(Send_{SP},price_1),\\ & (Send_{SP},price_1)(Rec_U,price_1)(Send_{SP},price_2),\\ & (Send_{SP},price_1)(Rec_U,price_2)(Send_{SP},price_1),\\ & (Send_{SP},price_2)(Rec_U,price_1)(Send_{SP},price_1),\\ & (Send_{SP},price_2)(Rec_U,price_1)(Send_{SP},price_2),\\ & (Send_{SP},price_2)(Rec_U,price_2)(Send_{SP},price_1),\\ & (Send_{SP},price_1)(Rec_U,price_2)(Send_{SP},price_2),\\ & (Send_{SP},price_2)(Rec_U,price_2)(Send_{SP},price_2)\}\end{aligned}$$

In general we have the requirement that in each group of actions that R knows to be correlated, it considers each of the parameters possible. In order to formalize this, we assign each group a number. We then built the μ-image of the sequences of actions that R considers possible after ω has happened, with the parameters being substituted by the number of the respective group they belong to. Then we check that when mapping these numbers arbitrarily onto possible parameters, this results in the μ-image of R's inverse view of ω, i.e. we check that the μ-image is (L, M)–complete for a specific language L and parameter set M.

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

Definition 1 *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*

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

In this definition, the set L consists of sequences of pairs *(action type,number)*. The functions $f \in M^{\mathbb{N}}$ map the numbers to parameter values in *M*. Therefore, $(i_{\Sigma_t}, f)(L)$ consists of sequences of pairs *(action type, parameter value)*. These sequences are in accordance with the correlations between parameter values defined by L. Now, K is (L, M)–complete if it consists of all possible sequences of pairs *(action type, parameter value)* derived by applying (i_{Σ_t}, f) to L for all possible mappings f from $\mathbb{N}$ to M.

This property allows the formalization of any of the above described situations. Let us consider as an example again the sequence of actions ω.

For the set $\{Send_{SP}, Rec_U\}$ of relevant action types we now choose a numbering that assigns the same number to those actions that are correlated:

$L_1 = \{(Send_{SP}, 1)(Rec_U, 1)(Send_{SP}, 2)\}$ Having chosen the language L_1 in this manner and considering the set $M = \{price_1, price_2\}$ of parameter values, $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ is (L_1, M)–complete if and only if

$$\begin{aligned}\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R) \quad = \quad & \{(Send_{SP}, price_1)(Rec_U, price_1)(Send_{SP}, price_1),\\ & (Send_{SP}, price_1)(Rec_U, price_1)(Send_{SP}, price_2),\\ & (Send_{SP}, price_2)(Rec_U, price_2)(Send_{SP}, price_1),\\ & (Send_{SP}, price_2)(Rec_U, price_2)(Send_{SP}, price_2)\}\end{aligned}$$

This exactly describes the situation in which R knows that the same price was received that was sent, but does not know which of the prices was sent. Note that if R considers more parameter values possible (i.e. if $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ contains more than the above four sequences), R still does not know which parameter was sent.

If R shall not know that there is a correlation between send and receive actions, the action types have to be numbered differently: no actions are correlated. This results in

$L_2 = \{(Send_{SP}, 1)(Rec_U, 2)(Send_{SP}, 3)\}$

The requirement of (L_2, M)–completeness results in the above mentioned eight sequences of pairs $(type, parameter)$. However, if R knows the correlation between $Send$ and Rec, i.e. if $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ contains only the four different sequences above (in which the same parameter value is sent and received), then $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ is not (L_2, M)–complete: Using $f(1) = price_1, f(2) = price_2$ and any $f(3)$ we obtain

$$(Send_{SP}, price_1)(Rec_U, price_2)(Send_{SP}, price_1) \notin \mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$$

Thus by appropriately numbering the action types, i.e. by appropriately choosing the language L, we can formalize which correlations between actions R is allowed to know, or in other words, which sequences of actions have to be included in W_R. This gives rise to the following definition:

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 R after ω with respect to (L, M)–completeness if there exists an (L, M)–complete language $K \subseteq (\Sigma_t \times M)^*$ with $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R) \supseteq K$.

Instead of separately defining different languages L for different sequences ω and the resulting set of sequences $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$, it is sufficient to have an appropriate L and an (L, M)–complete language $K \subseteq (\Sigma_t \times M)^*$ serving for all $\omega \in S$. Using the function p_1 that denotes the projection on the first component of a tuple, we introduce the following definition:

Definition 2 *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 $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R) \supseteq p_1^{-1}(p_1(\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R))) \cap K$ for each $\omega \in S$.*

Applying the projection p_1 and then the inverse p_1^{-1} to $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ results in sequences of actions where all parameter values occur and no grouping according to correlated actions has yet taken place. The intersection with the (L, M)–complete language K removes those sequences that do not match the respective grouping. Note that in the case of no correlation between actions, $p_1^{-1}(p_1(\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R))) \cap K$ $= p_1^{-1}(p_1(\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)))$.

More generally, the above equation can be used to define, for an arbitrary language $K \subseteq (\Sigma_t \times M)^*$ with $K \supseteq \mu(W_R)$, K–completeness of $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$. This allows to capture more sophisticated correlations between parameters.

3.2 A different property: M–rich

In some cases there is no adequate language L to describe that R cannot recognize the respective parameters used. We introduce a new example to motivate a weaker confidentiality property and we show that this weaker property may not be adequate for our previous price example. Let us consider a system which models an auction. In this system we look at the bidding phase. For simplicity we assume there are only two bidders *U1* and *U2*. The only possible action for bidders is *bid* with the parameters *bidder* and *amount*. In the same manner as above, agent *R* can observe the bidding actions using the action *obs*. We want to model the property that *R* may observe the amount which a bidder has made but is not allowed to know which bidder has made which bid. In contrast to (L,M)–completeness of the price in the previous section, *R* is allowed to know which bids have been made by the same bidder. For example, *R* may know that bids have been made alternately by two agents, but it is not allowed to know which bidder has started and which bidder has placed the winning bid. The homomorphism μ for this example can be defined as follows (for simplicity we neglect the values of the amount):

$$\begin{array}{lcl} \mu(bid(U, amount)) & = & (Bid, U) \\ \mu(obs(bid(U, amount))) & = & \varepsilon \end{array}$$

Now we consider the following sequence of actions:

$\delta =$ *bid(U2,amount$_1$) obs(bid(U2,amount$_1$)) bid(U1,amount$_2$) obs(bid(U1,amount$_2$)) bid(U2,amount$_3$) obs(bid(U2,amount$_3$)) bid(U1,amount$_4$) obs(bid(U1,amount$_4$))*

After having monitored δ, the following sequences of parameter values are possible in R's view of the system:

$$\begin{array}{lcl} \mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R) & = & \{(Bid, U2)(Bid, U1)(Bid, U2)(Bid, U1), \\ & & (Bid, U1)(Bid, U2)(Bid, U1)(Bid, U2)\} \end{array}$$

This knowledge of *R* corresponds to the confidentiality property that R cannot tell which bidder has placed which bid while knowing that $U1$ and $U2$ bid alternately. However, it is not possible to find a language L such that $\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R)$ is (L, M)–complete for $M = \{U1, U2\}$. Considering the correlations between actions known to *R* the following language $L3$ seems to be appropriate at first sight, as *R* knows that bidders place their bids alternately:

$L_3 = \{(Bid, 1)(Bid, 2)(Bid, 1)(Bid, 2)\}$

However, for the language L_3 chosen in this manner, $\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R)$ is not (L_3, M)–complete as $f(1) = f(2) = U1$ results in

$$(Bid, U1)(Bid, U1)(Bid, U1)(Bid, U1) \notin \mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R)$$

Nevertheless R does not know which of the two parameter values occured: for each of the bids it considers both bidders possible. To formalize this situation we need a property that does not consider the complete possible sequences of actions from R's local view but that considers only a "cut" through all sequences at the respective interesting actions. The following property describes the fact that from R's local view at each separate point in the sequence of actions, each of the parameter values is possible:

$$\forall u \in p_1(pre(\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R))) :$$
$$p_2(suf_1(p_1^{-1}(u) \cap pre(\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R)))) \supseteq M$$

Starting with the sequence of actions δ, we use λ_R^{-1}, λ_R and W_R to generate the set of possible sequences of actions that are identical to δ in R's local view. From these we extract, using the function μ, the relevant types with the respective parameter values, and pre generates all possible prefixes (i.e. we cut off the last action, the second last, etc.). With p_1 we disregard the parameters having been extracted by μ. Every u in a set of sequences generated in this manner is a sequence of types that correspond to those actions in δ that we are interested in, without the respective parameter values. p_1^{-1} again adds all parameter values in all possible combinations. The intersection with $pre(\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R))$ disregards those sequences that R does not consider possible because of its knowledge about correlation of actions. From each of the resulting sequences, we consider only the last element by applying suf_1 (where $suf_i(\omega)$ returns the suffix of ω with length i). p_2 then determines those parameter values that R considers possible in the respective action. The resulting set must include the complete set M of parameter values.

For the above example we get

$$p_1(pre(\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R))) = \{\varepsilon, Bid, BidBid, BidBidBid, BidBidBidBid\}$$

$u = BidBid$ for example results in

$$\begin{aligned} p_1^{-1}(u) = \quad & \{(Bid, U1)(Bid, U1), (Bid, U1)(Bid, U2), (Bid, U2)(Bid, U1), \\ & (Bid, U2)(Bid, U2)\} \end{aligned}$$

We now intersect this set with the set of sequences of types (with parameters) that R considers possible as described above.

$$p_1^{-1}(u) \cap pre(\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R)) = \{(Bid, U1)(Bid, U2), (Bid, U2)(Bid, U1)\}$$

suf_1 reduces the resulting $(type, parameter)$ sequences to the respective last $(type, parameter)$ (that is, to (Bid, Ui)), and finally p_2 extracts the parameters that R considers possible in this $(type, parameter)$. If this set does not include M completely then R knows more about the parameters that are possible in this action than it should know after having monitored δ. In our example, $p_2(suf_1(\{(Bid, U1)(Bid, U2), (Bid, U2)(Bid, U1)\}))$ $= \{U1, U2\} = M$.

Analogously to definition 1 we give the following general definition:

Definition 3 *For a given set M of parameter values and a set Σ_t of action types we call the language $K \subseteq (\Sigma_t \times M)^*$ M–rich if*

$$\forall u \in p_1(pre(K)) : p_2(suf_1(p_1^{-1}(u) \cap pre(K))) \supseteq M$$

Analogously to (L, M)–completeness, we can now define a different kind of parameter confidentiality:

Definition 4 *Let M be a parameter set, Σ a set of actions, Σ_t a set of types, and $\mu : \Sigma^* \to (\Sigma_t \times M)^*$ a homomorphism. Then M is parameter-confidential for R with respect to M–richness if $\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R)$ is M–rich for all $\omega \in S$.*

For the price-offer example explained in section 3.1, the property M–rich may be too weak. Consider for example a case with only two possible prices and *SP* offering alternately the two different prices to *U*. Now, if $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ is M–rich, *R* cannot tell which price has been offered in which action. However, as *R* can observe that two prices have been offered alternately, *R* can calculate the average price offered to *U*, which may be undesirable.

For a given $type$ sequence, (L, M)–completeness of a language $K \subseteq (\Sigma_t \times M)^*$ exactly determines the set of $(type, parameter)$ sequences which have to be in K. This is not true in the case of M–richness: As mentioned above, the equation

$$\begin{aligned} \mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R) \quad = \quad & \{(Send_P, m1)(Rec_Q, m1)(Send_P, m2), \\ & (Send_P, m2)(Rec_Q, m2)(Send_P, m1)\} \end{aligned}$$

implies M–richness of $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$. But also the equation

$$\begin{aligned} \mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R) \quad = \quad & \{(Send_P, m1)(Rec_Q, m1)(Send_P, m1), \\ & (Send_P, m2)(Rec_Q, m2)(Send_P, m2)\} \end{aligned}$$

would imply M–richness of $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$. So two different languages K express the required parameter variety.

Therefore, to bridge the formal gap between (L, M)–completeness and M–richness we need to consider the family $\mathcal{K}$ of all languages $K \subseteq (\Sigma_t \times M)^*$ which are M–rich. Now $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ is M–rich if and only if $\exists K \in \mathcal{K} : \mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R) = p_1^{-1}(p_1(\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R))) \cap K$.

4 Correlation between the two different properties

It can be shown that (L, M)–completeness of the language $\mu(\lambda_R^{-1}(\lambda_R(\omega)) \cap W_R)$ implies its M–richness (see Theorem 1 below). However, the reverse statement does not hold in general. Yet, if considering only one action when applying μ, both properties are equivalent (see Theorem 2). We again use the bidding example to illustrate this. Let us assume we are only interested in the last bidding action of δ (i.e. $\Sigma' = \{bid(U1, amount_4)\}$) and the system shall guarantee that R does not know who of the two agents placed the bid. Trivially, the only language L to choose is $L = \{(Bid, 1)\}$. Thus, (L, M)–completeness implies $\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R) \supseteq \{(Bid, U1), (Bid, U2)\}$.

In other words, the image of μ contains sequences of types with parameters, each of the sequences having only one element. On the other hand, M–richness implies that $p_2(suf_1(\{(Bid, U1), (Bid, U2)\} \cap \mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R) \supseteq M$, which is the case if $\mu(\lambda_R^{-1}(\lambda_R(\delta)) \cap W_R) \supseteq \{(Bid, U1), (Bid, U2)\}$.

Theorem 1 *For prefix closed languages $K \subseteq (\Sigma_t \times M)^*$ and $L \subseteq (\Sigma_t \times \mathbb{N})^*$, (L, M)–completeness of K implies M–richness.*

Proof: Let us assume that K is an (L, M)–complete language which is not M–rich. Thus there exists $u \in p_1(pre(K))$ with $p_2(suf_1(p_1^{-1}(u) \cap pre(K))) \subsetneqq M$, i.e. $u \in p_1(K)$ with $p_2(suf_1(p_1^{-1}(u) \cap K)) \subsetneqq M$ (K is prefix closed). Hence there exists $n \in M, n \notin p_2(suf_1(p_1^{-1}(u) \cap K))$ which means that none of the sequences in K that correspond to u with respect to the type (generated by p_1) contains n as parameter in the last element. Let us further consider $v \in L$ that equals u with respect to the type components, i.e. $p_1(v) = p_1(u)$, and the map $r : \mathbb{N} \longrightarrow M$ with $r(x) = n$ for all $x \in \mathbb{N}$. Clearly, as in particular the last element of $(i_{\Sigma_t}, r)(v)$ contains the parameter n as the second component (all elements contain this parameter), and as n is not contained as parameter of the last element in any of the sequences in K corresponding to v, $(i_{\Sigma_t}, r)(v) \notin K$. But then K is not (L, M)–complete.

Theorem 2 *For a set $K \subseteq (\Sigma_t \times M)$ and $L \subseteq (p_1(K) \times \mathbb{N})$, M–richness of K implies (L, M)–completeness.*

Proof: Let $K \neq \emptyset$ (the statement is trivial for the empty set) and $u \in p_1(K)$. Because of $l(u) \leq 1$, $p_1(pre(K)) = p_1(K) \cup \{\varepsilon\})$ holds. Furthermore $suf_1(p_1^{-1}(p_1(u))) = p_1^{-1}(p_1(u))$ and $p_2(suf_1(p_1^{-1}(p_1(u)))) = M$. This implies that $p_2(suf_1(p_1^{-1}(p_1(u)) \cap pre(K)) = M$ if and only if $p_2(p_1^{-1}(p_1(u)) \cap K) = M$.
Let now $f \in M^{\mathbb{N}}$. For each $v \in L$, $p_1(v) \in p_1(K)$ holds because of $L \subseteq (p_1(K) \times \mathbb{N})$. Thus, for each $v \in L$, there is an $w \in K$ with $p_1(w) = p_1(v)$. As K is M–rich, $p_2(suf_1(p_1^{-1}(p_1(u)) \cap pre(K)) = M$, thus $p_2(p_1^{-1}(p_1(w)) \cap K) = M$. It follows $(i_{\Sigma_t}, f)(v) \in K$ for all $f : \mathbb{N} \longrightarrow M$. Hence K is (L, M)–complete.

5 Conclusion

We have introduced the new notion of parameter-confidentiality in terms of formal languages. In contrast to previous definitions, confidentiality of certain parameters can be specified relative to an agent's knowledge about the system, especially about dependencies between parameter values in different actions. A wide variety of confidentiality properties for communicating systems can be exactly specified using the two definitions of (L,M)–completeness and M–richness. The universality of our formal definitions allows to apply them to any specification language with a semantics based on labeled transition systems. Parameter-confidentiality complements existing concepts of non-interference, information flow, secrecy and indistinguishability.

The definitions introduced in this paper fit in our design method for security sensitive systems, where security properties are specified independently from the abstraction level. Suitable language homomorphisms map from lower to higher levels of abstraction. Our design method is successfully applied in the project CASENET funded by the European Commission (IST-2001-32446), where it is used to develop real life applications with certain security properties. In a forthcoming paper, conditions on homomorphisms under which they preserve parameter-confidentiality will be presented.

References

[AG99] M. Abadi and A Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, 1999.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In H. Krawczyk, editor, *Advances in Cryptology - Crypto 98*, Lecture Notes in Computer Science, pages 26–45. Springer Verlag, 1998.

[Eil74] S. Eilenberg. *Automata, Languages and Machines*. Academic Press, New York, 1974.

[GM82] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[GO01] R. Grimm and P. Ochsenschläger. Binding Cooperation, A Formal Model for Electronic Commerce. *Computer Networks*, 37:171–193, 2001.

[GOR02] S. Gürgens, P. Ochsenschläger, and C. Rudolph. Authenticity and provability, a formal framework. In *Infrastructure Security Conference InfraSec 2002*, volume 2437 of *Lecture Notes in Computer Science*, pages 227–245. Springer Verlag, 2002.

[Man00] H. Mantel. Possibilistic Definitions of Security – An Assembly Kit. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 185–199, 2000.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Rud01] C. Rudolph. *A Model for Secure Protocols and its Application to Systematic Design of Cryptographic Protocols*. PhD thesis, Queensland University of Technology, 2001.