Confidentiality-Preserving Refinement

Maritta Heisel
Fakultät für Informatik
Universität Magdeburg
39016 Magdeburg
Germany
heisel@cs.uni-magdeburg.de

Andreas Pfitzmann
Fakultät für Informatik
Technische Universität Dresden
01062 Dresden
Germany
pfitza@inf.tu-dresden.de

Thomas Santen
Fachbereich Informatik
Technische Universität Berlin
10587 Berlin
Germany
santen@acm.org

Abstract

We develop a condition for confidentiality-preserving refinement which is both necessary and sufficient. Using a slight extension of CSP as notation, we give a toy example to illustrate the usefulness of our condition.

Systems are specified by their behavior and a window. For an abstract system, the window specifies what information is allowed *to be observed by its environment. For a concrete system, the window specifies what information* cannot *be hidden from its environment. A concrete system is a confidentiality-preserving refinement of an abstract system, if it behaviorally refines the abstract system and if the information revealed by the concrete window is allowed to be revealed according to the abstract window.*

1 Introduction

Dependable IT-systems [10] of relevant size can only be built if all dependability attributes have a clear meaning. This meaning has to be consistent both with the common understanding of the people building and using the IT-system as well as with the tools they use and the development process they adhere to. Therefore, a formal meaning of all dependability attributes is needed which is compatible with the usual refinement process of systems engineering.

Our goal is to establish a formal meaning of security attributes. Confidentiality, integrity, and availability are seen as the generic properties of security [20, 3]. Some authors propose to add accountability as a fourth one [2], others propose to refine confidentiality, integrity and availability according to the kind of information they relate to [21]. Then, e.g. accountability can be interpreted as an integrity property concerning the circumstances of an action, anonymity can be interpreted as a confidentiality property concerning the circumstances, and so on.

The relationship between the security attributes integrity and availability, and formal notions compatible with refinement is roughly clear. For terminating computations, integrity corresponds to partial correctness and availability corresponds to assured termination combined with sufficient computational resources to fulfill real-time requirements. Integrity and availability together correspond to total correctness and sufficient computational resources. For reactive systems, integrity means that the defined processes satisfy certain required predicates, and availability corresponds to fairness and liveness combined with sufficient computational resources.

To date, the relationship between confidentiality and refinement is less well understood. This is our motivation to develop a formal notion of a confidentiality-preserving refinement. This notion must be consistent with all properties of refinement, which mainly provide for integrity and partially for availability. We must add requirements which guarantee that each refinement step demonstrably preserves all relevant confidentiality properties. To allow for stepwise refinement, the definition of refinement to be developed must be transitive: If system 2 is a confidentiality-preserving refinement of system 1 and system 3 is a confidentiality-preserving refinement of system 2, then system 3 must be a confidentiality-preserving refinement of system 1.

When specifying a secure system, we do not only define its functionality, but also specify what information about it its environment may observe. A refinement of a system consists of a functional description and the specification of what information its environment can possibly observe. The refinement is confidentiality-preserving if the more concrete system conveys only information to its environment about the data represented in the abstract system that the abstract system allows the environment to observe.

In describing the refinement relation between the more abstract and the more concrete system, we do not only define relations between abstract and concrete data and behavior, but we also consider probabilistic behavior of the involved systems. We derive a condition on these probabilities, which is both sufficient and necessary for preserving confidentiality.

The paper is organized as follows: Section 2 discusses different approaches to formally capture confidentiality and to preserve confidentiality in refinements. Section 3 briefly describes an extension of CSP with probabilistic choice and the concept of behavioral refinement. In Section 4, we describe how systems with confidentiality requirements are specified. The notion of indistinguishability, which is the basis of our confidentiality property, is introduced in Section 5. In Section 6, we state our main definition – confidentiality-preserving refinement – and show the transitivity of the definition. All our definitions are illustrated by a running example. In Section 7, we summarize our contributions and point out directions for further research.

2 Formal Approaches to Confidentiality and Refinement

Non-interference [4] is a security property, which has been studied extensively and has been formally treated using CSP [6], see for example the work of Allen [1], Graham and Cumming [5], and Roscoe, Woodcock and Wulf [13]. Given a system S and two users u and v, non-interference states that v's view of S is completely unaffected by u's actions. Non-interference thus guarantees that no information can flow from u to v. Roscoe et. al. [13] have shown how non-interference can be preserved by refinement for deterministic systems.

Mantel [11] considers the preservation of information flow properties under refinement. It is well-known that CSP-style refinement does not preserve information flow properties in general [7]. Mantel shows how refinement operators tailored for specific information flow properties can modify an intended refinement such that the resulting refinement preserves the given flow property. Working top-down from the specification to an implementation, the refinement operators may lead to concrete specifications that are practically hard to implement, because the changes in the refinement they induce are hard to predict and may not be easy to realize in an implementation.

Our way of describing the confidentiality properties of an implementation is independent of the refinement and – in that respect – we work more in a bottom-up rather than in a top-down fashion. In contrast to Mantel's approach, where the refinement operators partly determine the implementation, in our approach, it is the developers' responsibility to construct the implementation in such a way that it preserves confidentiality. A further difference to Mantel's setting is that we consider data refinements whereas he works with atomic events only. Also, Mantel does not consider the probabilistic behavior of systems but works in a possibilistic setting.

In general, a specification is a model of the system to be implemented and as such it is an abstraction of the implementation (e.g. it does not make statements about resources). It is unavoidable that an adversary can distinguish more data in an implementation than the data about which a specification can even make statements. When considering confidentiality-preserving refinement, it is therefore necessary to find a way of preventing an adversary from gaining information about confidential data even if s/he derives information from data which cannot be expressed in the specification.

Jürjens [9] pursues a goal similar to ours. He distinguishes two kinds of non-determinism. The first kind corresponds to under-specification, leaving room for implementation decisions. This kind of non-determinism may be eliminated in a refinement. The second kind of non-determinism serves to protect secrets and is not meant to be eliminated in a refinement. Jürjens correctly observes that the distinction between those kinds of non-determinism has previously been blurred in formal approaches to define security properties, which has lead to anomalies in the resulting theories.

Jürjens uses stream processing functions to model systems. He defines a notion of secrecy which is only necessary in the sense that it does not prevent implicit information flows. His condition is also only necessary because an observing adversary gains a secret either completely or not at all. Jürjens does not consider that an adversary may gain information about the secret in a probabilistic sense. He identifies conditions under which certain refinement operators on stream processing functions preserve his notion of secrecy.

In Jürjens' approach, the secrets to be kept confidential must be expressed explicitly. Hence, omissions in a specification may lead to an insecure system. The approach we present in this paper explicitly specifies what is allowed to be observed – all other data are to be kept confidential.

There is a variety of formal notions of confidentiality, ranging from very strong sufficient conditions such as non-interference (which are not necessary) to weaker necessary conditions such as the one of Jürjens [9] (which are not sufficient). We come up with a *characterization* of confidentiality-preserving refinement, i.e., with a condition being both necessary and sufficient. To be able to state such a condition, we take probabilistic mechanisms into account, whose indispensability is acknowledged by the security community, but which are missing in most formal treatments of the subject.

An exception is Toussaint's work on protocol verification [17, 18, 19]. Augmenting protocol states by "known" and "believed" terms of an algebra of encryption and communication, she models the knowledge of the participants in a cryptographic protocol. Considering transition probabilities between protocol states, she can analyze whether probabilistic constraints make protocols secure [17].

Toussaint also separates the protocol proper from the cryptographic system used to "implement" the protocol [19], which allows her to analyze protocols independently of the properties of specific encryption algorithms. Her notion of an "implementation", however, covers only a small fraction of ours, because we consider the concrete choice of data representations (of plain text, cipher text, keys, etc.) in addition to the choice of a cryptographic algorithm.

In contrast to the approaches discussed so far, Jacob [8] does not define an "absolute" confidentiality property that is either fulfilled by a given system or not. Instead, he considers varying *degrees* of confidentiality. To be able to compare confidentiality, he uses *windows*, which are defined to be sets of atomic interactions. A system is more confidential than another with respect to a given window if it allows less interactions on that window. For a refinement, Jacob requires that each window's "view" does not change. As before, this definition does not capture a quantitative gain of information.

We have taken up the idea of using a window that allows us to define selective views on a given system, which may be attributed to an observer or an adversary. However, in our approach, a window will be a communication channel to which all data an observer can gather are written. Everything that cannot be distinguished using the data made available by the window channel is hence kept secret by the system.

3 Formal System Specification and Behavioral Refinement

On a conceptual level, the definition of confidentiality-preserving refinement that we develop in this paper is independent of a particular formalism. For illustrative purposes, we use the process specification language CSP [6] to specify the behavior of systems. We are aware of the fact that certain relevant properties of systems that may affect confidentiality, such as real-time behavior and use of computational resources, cannot be expressed in CSP. Choosing a different formalism to capture those properties would complicate the presentation but would not change the conceptual argument that underlies our definitions.

CSP is a process calculus. A *process* Q has an *alphabet* αQ that denotes the set of names of the channels over which Q communicates with the outside world. An *event* $c.v$ describes an instance of communicating the value v over the channel c.

There are several variants of semantics of CSP processes. All are based on the notion of the *traces* of a process Q, $\mathit{traces}(Q)$. A trace of Q is a sequence of events in which the process can engage. After Q has engaged in a trace s, it may refuse to engage in a number of events. The set X of those events is called the *refusals* of the process Q/s, which is Q after performing s. The set of *failures* of Q, $\mathit{failures}(Q)$, contains all pairs (s, X) where $s \in \mathit{traces}(Q)$ and X the set of refusals of process Q/s.

The failure semantics of CSP distinguishes more processes than the trace semantics. There is a third, even finer semantics of CSP, called the failure-divergence semantics. It distinguishes certain infinite behaviors in which we are not interested. We therefore use the failure semantics.

3.1 Probabilistic Processes

The internal choice operator of CSP serves to specify *nondeterministic* processes: $A \sqcap B$ is a process that either behaves like A or like B; the environment has no means to influence that choice. No statement is made about the probability with which the process exhibits the behavior of A or of B.

In the context of secure systems, it is necessary to distinguish that form of non-determinism from a *probabilistic* one (an observation we share with Jürjens [9]): the probability of choosing A or B may serve an adversary to infer information about confidential data in the system. To model such a situation, we extend the notation of CSP by a *probabilistic choice* operator that is parameterized by a probability distribution. The process $\bigsqcap_k^P Q(k)$ chooses a value for k according to the probability distribution P; then it behaves like $Q(k)$, where Q is a family of processes indexed by k.

The probabilistic choice must be reflected in the semantics: the failures of $\bigsqcap_k^P Q(k)$ with a probability greater than zero are failures of the usual nondeterministic choice $\bigsqcap_k Q(k)$; in addition, a probabilistic choice induces a family of probability distributions P_Q^n on the failures of a process Q: for each n, P_Q^n is a distribution on the failures containing traces of length n. To keep our notation concise, we will use the term "distribution on the failures of Q", denoted P_Q, when referring to traces of the same length.

3.2 Behavioral Refinement

In the following definition of behavioral refinement, we disregard the probabilistic behavior of processes and treat the probabilistic choice as if it were a usual nondeterministic choice. We will consider the probabilistic aspects of process refinement when we define confidentiality-preserving refinement in Section 6.2.

The notion of *refinement* between processes describes that a process B "implements" the behavior of process A, written $A \sqsubseteq B$. Refinement in CSP is usually defined as set inclusion on the chosen semantics, i.e. for the failure semantics, we get that process B refines process A if $\mathit{failures}(B) \subseteq \mathit{failures}(A)$. Process B is then called the *concrete*, process A the *abstract* process of that refinement.

We are interested in a generalization of that notion that makes it possible to consistently replace data in events. Replacing data is necessary because confidentiality-preserving refinement as we define it in Section 6 not only allows implementors to refine the behavior of a system but also to refine the concrete data that it uses to communicate with the outside world. A *retrieve relation* R models that substitution of data. It relates the traces of two processes A and B with identical alphabets, $\alpha A = \alpha B$, by consistently replacing events $c.w$ of B by events $c.v$ of A. Thus, a retrieve relation maps concrete traces to abstract ones.

We say that a process A is *behaviorally refined* by process B if there exists a retrieve relation R such that the set of abstractions of the failures of B with respect to R is contained in the set of failures of A. Formally, we write

$$A \sqsubseteq_R B \Leftrightarrow R(\!|\ \mathit{failures}(B)\ |\!) \subseteq \mathit{failures}(A)$$

where $R(\!|\ D\ |\!)$ maps the set D to the union of the sets of images of members of D under R. We assume the obvious extension of R from traces to failures. We indicate a suitable retrieve relation R as an index to the refinement symbol, because we often need to refer to it in proofs.

4 System and Adversary Specification

The basis of our approach is to augment a specification of the intended behavior of a system – given in terms of a CSP process – by a *window* that models the possible observations an adversary may make about the system. The window is a distinguished channel to which the system writes data. These data can be used by the environment and be further processed without any restriction. We assume an open system design process that does not try to realize “security by obscurity”. Therefore, we must assume the adversary to know the structure and behavior of the system. The adversary may use that knowledge to derive information about the internal state of the system from the observations at the window. Specifying the behavior of the system on the window channel thus completely describes the information an adversary can obtain by observing the system. A system specification consists of the process describing the system behavior and the window channel.

Definition 1 (System, Window)
A system specification $S = (Q, w)$ is a pair of a process definition Q and a distinguished channel $w \in \alpha Q$, called the window *of S.*

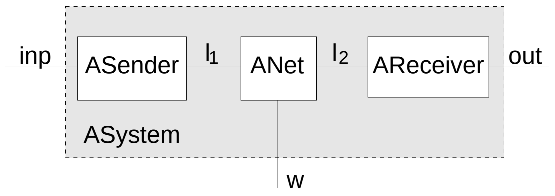

Figure 1. The abstract system

In our presentation, we consider only one adversary. Differentiating between adversaries with different capabilities would mean to introduce distinct windows for each adversary.

Example. We illustrate our approach by the example of a communication between two parties via an untrusted channel. Figure 1 illustrates the following CSP specification of an abstract view of that system. Events on the channels *inp* and *out* model the data that the sender process *ASender* transmits over the untrusted network *ANet* to the receiver process *AReceiver*. The internal channels l_1 and l_2 serve to describe the communication between the three processes. The events on those channels are not visible from outside the system.

The purpose of the window w at this stage of the specification is to describe what information an external observer (malicious or not) *may* obtain about the communication between sender and receiver. We decide that an observer is allowed to see the *length* (in whatever suitable measure) of the messages exchanged between sender and receiver – but nothing else.

Formally, we specify *ASystem* as a CSP process that is the parallel composition of the three processes *ASender*, *ANet*, and *AReceiver*. The first two communicate via channel l_1 and the last two via channel l_2. Hiding l_1 and l_2, we make the events taking place on those channels internal to the system.

$$\mathit{ASystem} \mathrel{\widehat{=}} (\mathit{ASender} |[l_1]| \mathit{ANet} |[l_2]| \mathit{AReceiver}) \setminus \{l_1, l_2\}$$

The behavior of *ASender* and *AReceiver* is simple: *ASender* writes whatever data *msg* it receives on the input channel *inp* to the channel l_1; similarly, *AReceiver* copies all data it receives on l_2 to *out*.

$$\mathit{ASender} \mathrel{\widehat{=}} \mathit{inp}?\mathit{msg} \rightarrow l_1!\mathit{msg} \rightarrow \mathit{ASender}$$

$$\mathit{AReceiver} \mathrel{\widehat{=}} l_2?\mathit{msg} \rightarrow \mathit{out}!\mathit{msg} \rightarrow \mathit{AReceiver}$$

The network *ANet*, however, not only copies the data *msg* from l_1 to l_2, but it also writes the length of each received data item to the window w. Thus we model the information the network conveys to the outside world about the communication taking place between sender and receiver.

$$ANet \mathrel{\widehat{=}} l_1?msg \rightarrow w!length(msg) \rightarrow l_2!msg \rightarrow ANet$$

5 Indistinguishability

Inferring information about a system (Q, w) through the window w means to distinguish data the system processes by different observations on w. Conversely, to keep information confidential, the system must be designed in such a way that the data the window provides to an observer cannot be used to distinguish data it internally stores and that it should keep confidential. In other words, the window of a system specifies the confidentiality property of the system: its "secret" is given by the data that are indistinguishable by observing the window only.

Formally, indistinguishability is an equivalence relation on the traces of a system. Two system traces cannot be distinguished by the environment if their projections to the window are the same. Considering just single data items that appear on the window would be insufficient, because an adversary might accumulate information by observing the window for a longer time.

Definition 2 (Indistinguishability)
Let $S = (Q, w)$ be a system specification. Let win be the function projecting traces in $traces(Q)$ to traces of $Q \setminus (\alpha Q - \{w\})$, i.e. to the sequences of events on the window w. Two traces $s, t \in traces(Q)$ are indistinguishable *iff the projections to w are equal:*

$$s \equiv t \Leftrightarrow win\, s = win\, t$$

Example. The possible traces of *ASystem* are given by sequences of events on the channels *inp*, *w*, and *out*: some data item *msg* is input on *inp*, its length is written to *w*, and it is output to *out*. Then the system recurs and produces a similar sequence of events for another data item.

$$traces(ASystem) = \\ \{msg \bullet \langle inp.msg, w.length(msg), out.msg\rangle\}^*$$

We use the notation of set comprehension known from Z [15]: The comprehension $\{x, y, z \mid P(x, y, z) \bullet t(x, y, z)\}$ denotes the set of all $t(x, y, z)$ for which there exist x, y, and z such that the predicate $P(x, y, z)$ holds; if $P(x, y, z)$ is universally true or if $t(x, y, z) = (x, y, z)$, then we write $\{x, y, z \bullet t(x, y, z)\}$ and $\{x, y, z \mid P(x, y, z)\}$, respectively. For a set of sequences T, the set T^* is the set of all possible concatenations of members of T.

Knowing what the traces of *ASystem* are, we can derive a condition characterizing their indistinguishability. Two traces $s, t \in traces(ASystem)$ cannot be distinguished by observing w if they are of equal length and the lengths of corresponding messages are equal.

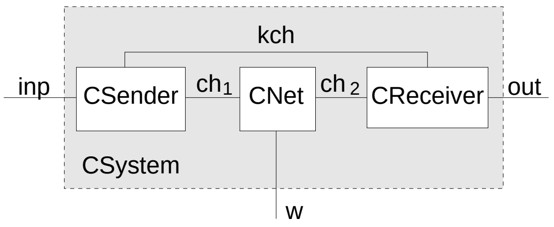

Figure 2. The concrete system

$$\begin{aligned} s \equiv_a t \quad &\Leftrightarrow \quad win\, s = win\, t \\ &\Leftrightarrow \quad \#s = \#t \land \\ &\qquad \forall\, i : \operatorname{dom} s \bullet \\ &\qquad\quad length(Msg(s, i)) = length(Msg(t, i)) \end{aligned}$$

The domain $\operatorname{dom} s$ of a sequence s is the set of indexes $\{1, 2, 3, \ldots \#s\}$, where $\#s$ is the length of s. The function $Msg(s, i)$ denotes the data *msg* that appeared most recently in an event *inp.msg* before or at position i in s, i.e. at the maximal index $j \leq i$ where $s(j)$ is an event at channel *inp*. Hence, $Msg(s, i)$ is the data item the system processes at the i-th event of s.

In our example, the equality $win\, s = win\, t$ implies that the sequences s and t have the same length. In general, this is not true, because different internal behavior with a different number of events that are not visible at the window channel may still result in the same projection to the window channel.

6 Confidentiality-Preserving Refinement

We now introduce the main contribution of this paper: confidentiality-preserving refinement. By way of motivation, we extend our example and describe a concrete system that we consider one step in refining *ASystem* to an implementation. Then, we generalize our observations to define confidentiality-preserving refinement formally.

Example. Figure 2 shows *CSystem*, which we wish to use as an implementation of *ASystem*. With respect to the interface, both systems are very similar: we again have the three channels *inp*, *out*, and *w*. The data transmitted from *inp* to *out* shall remain the same, but the data on *w* changes with the transition from *ASystem* to *CSystem*: In *ASystem*, we had a quite abstract view on the network, which allowed us to express the confidentiality property that only the length of transmitted messages may be observed by the outside world. In *CSystem*, we now consider a more realistic model that will expose the full data transmitted over the network to an external observer. Therefore, we specify the network process *CNet* so as to copy all data *ct* it receives unchanged to the window *w*.

$$CNet \mathrel{\widehat{=}} ch_1?ct \to w!ct \to ch_2!ct \to CNet$$

Obviously, to use such a network for confidential communication that reveals only the length of messages but not their content – as specified in *ASystem* – we need to encrypt the transmitted data. We introduce a channel *kch* between *CSender* and *CReceiver* that allows them to exchange keys in a secure way. To transmit a data item *msg*, *CSender* first chooses a key *k* randomly according to some probability distribution *P*. *CSender* transmits *k* to *CReceiver* via *kch* and then transmits the ciphertext $cipher(msg, k)$ over the network. Using the same key *k*, *CReceiver* decrypts the ciphertext it receives from the network.

$$\begin{aligned} CSystem \;&\widehat{=}\; (CSender |[kch]| CReceiver \\ &\qquad |[ch_1, ch_2]| CNet) \setminus \{kch, ch_1, ch_2\} \end{aligned}$$

$$\begin{aligned} CSender \;&\widehat{=}\; \textstyle\bigsqcap_k^P (kch!k \to inp?msg \to \\ &\qquad ch_1!cipher(msg,k) \to skip); \\ &\quad CSender \end{aligned}$$

$$\begin{aligned} CReceiver \;&\widehat{=}\; kch?k \to ch_2?ct \to \\ &\qquad out!decipher(ct,k) \to CReceiver \end{aligned}$$

To keep the example simple, *CSender* synchronously transmits keys and ciphertexts. A further refinement would transmit a number of keys in advance and only later use those keys for encryption. This would require a more complex synchronization between *CSender* and *CReceiver*.

What information can an observer obtain about the communication between *CSender* and *CReceiver*? As for the abstract system, the traces of *CSystem* consist of recurring data input on *inp*, and outputs to *w* and *out*. The data copied to *w* is the ciphertext for the data *msg* and some key *k*.

$$\begin{aligned} & traces(CSystem) = \\ &\qquad \{msg, k \bullet \langle inp.msg, w.cipher(msg,k), out.msg\rangle\}^* \end{aligned}$$

The distribution $P_{CSystem}$ on $traces(CSystem)$ is determined by *P*: each key *k* used in a trace $t \in traces(CSystem)$ is independently chosen according to *P*. Thus, with the sequence $K(t) = \langle k, msg, i \mid i \in \operatorname{dom} t \wedge t(i) = w.cipher(msg,k) \bullet k\rangle$ of keys used in *t*, for given plaintexts $Msg(t,i)$, the probability $P_{CSystem}(t)$ that the system performs *t* is the product of the probabilities to choose the keys in $K(t)$:

$$P_{CSystem}(t) = \Pi_{i \in \operatorname{dom} K} P(K(t)(i))$$

An observer cannot distinguish two traces $s, t \in traces(CSystem)$ if they are equally long and for each data communication the keys used for that communication map the possibly different plain texts to identical ciphertexts.

$$\begin{aligned} s \equiv_c t \;&\Leftrightarrow\; win\, s = win\, t \\ &\Leftrightarrow\; \#s = \#t \wedge \\ &\qquad \forall\, i : \operatorname{dom} s \bullet \\ &\qquad\quad cipher(Msg(s,i), Key(s,i)) = \\ &\qquad\quad\; cipher(Msg(t,i), Key(t,i)) \end{aligned}$$

The function $Key(s,i)$ returns the key *k* used to encrypt $Msg(s,i)$. The events in *s* do not contain *k* explicitly, but only the plaintext $Msg(s,i)$ and its corresponding ciphertext. To keep the example simple, we assume that the function *cipher* is invertible in its second argument such that *k* is uniquely determined by *msg* and $cipher(msg,k)$.

Neither the messages $Msg(s,i)$ and $Msg(t,i)$ nor the keys $Key(s,i)$ and $Key(t,i)$ need to be equal. At this stage of the argument, we cannot deduce how much information an adversary actually obtains by observing ciphertexts at *w*. This depends on the properties of the encryption function *cipher*.

6.1 Preservation of Indistinguishability

In the transition from an abstract to a concrete system specification, the interpretation of a window changes: The window of an *abstract* system specifies what information is *allowed* to be visible to the outside world. The window of a *concrete* system specifies what information *cannot* be hidden from the environment. In the following, we present a necessary and sufficient condition for a refinement to preserve confidentiality.

The concrete system must not convey more information through its window to the environment than specified in the abstract window. How can we make that intuition precise? Two indistinguishable traces of the concrete system do not convey any information about the differences of internal data in the system to the environment. But how much information do two *distinguishable* traces of the concrete system provide about data that shall be kept confidential, i.e. indistinguishable, according to the abstract system specification?

Here, a purely logical argument – that many approaches to formally describe secure systems prefer, see Section 2 – is insufficient, because it is not enough to ask whether a distinction in the concrete system *definitely* allows an observer to distinguish confidential data, but we must describe whether such a distinction provides *more* information[1] about the confidential data than the abstract window reveals. Therefore, we consider the respective probabilities of internal data that may cause a particular observable behavior on a window. Figure 3 illustrates our approach to formalizing that probabilistic argument:

Let *r* and *s* be two abstract traces that are indistinguishable with respect to the window. According to *R*, trace *r*

[1]Note that information is a probabilistic notion [14].

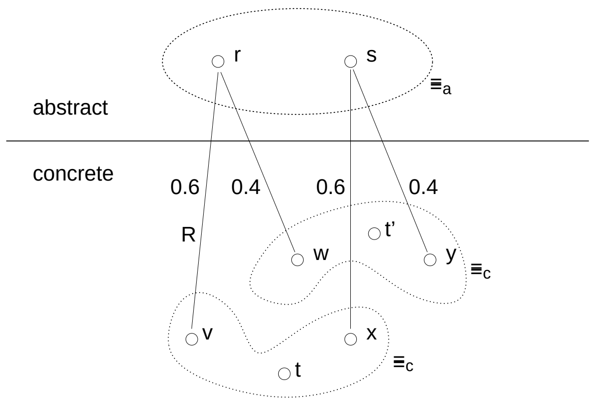

Figure 3. Probabilistic concretization and indistinguishability

can be represented by the concrete traces v and w, and trace s can be represented by the concrete traces x and y, where v and x as well as w and y are indistinguishable by observing the concrete window. For keeping r and s indistinguishable in the concrete system, we must require that the probability that r is represented by v be the same as the probability that s is represented by x. If this were not the case, an adversary might be able to gain information whether r or s happened on the abstract layer: if the probability that r is represented by v is greater than the probability that s is represented by x, for an adversary, the observation of some element $t \equiv_c v$ increases the probability of r with respect to s. Therefore, the condition just described and formally expressed in Definition 3 is *necessary*.

On the other hand, the condition also is *sufficient*: Suppose that an adversary is only able to distinguish the sets $\{v, x\}$ and $\{w, y\}$ as runs of the concrete system. Further suppose that the probability that the concrete system chooses v to implement r is the same as the probability that the concrete system chooses x to implement s. Then the observation of a behavior indistinguishable from v and x does not convey any information to the adversary whether the abstract system performs r or s. The same is true for the adversary observing the set $\{w, y\}$.

The fact that there are different probabilities of representing an abstract trace by a concrete one are caused by non-determinism in the concrete system that is used to provide confidentiality. In our example, probabilistically choosing the key to encrypt data is the source of that kind of non-determinism. In the relation between the abstract and the concrete system, probabilistic non-determinism is reflected by different possibilities to represent abstract data on the window (“lengths” in our example) by concrete data (ciphertexts). The concrete system probabilistically chooses one of those possibilities. The distribution of that choice must satisfy the restrictions on indistinguishability we just discussed. This insight is the key to defining confidentiality-preserving refinement.

6.2 Refinement

Each confidentiality-preserving refinement must of course be a correct behavioral refinement. For two systems $A = (Q_a, w)$ and $C = (Q_c, w)$, we must ensure $Q_a \sqsubseteq_R Q_c$ for some retrieve relation R. Moreover, on the concrete level, it must not be possible to distinguish more traces than on the abstract level via the respective windows. For two indistinguishable abstract traces $r, s \in traces(Q_a)$, this means that the probability of representing r by a concrete trace $u \in R^{-1}(r)$ that is indistinguishable from a given trace t must be equal to the probability of representing s by such a trace. The probabilities of those representations are determined by the distribution of the traces of system C. Phrased more precisely, the probability of representing r by u is the probability that process Q_c chooses u among the ones in $R^{-1}(r)$ under the condition that Q_c simulates the abstract behavior r.

Definition 3 (Confidentiality-Preserving Refinement) *Let $A = (Q_a, w)$ and $C = (Q_c, w)$ be two system specifications. Let $\equiv_a$ be the observational equivalence in A (wrt. w), $\equiv_c$ be the observational equivalence in C, and let P_c be the probability distribution on $traces(Q_c)$. The system C is a* confidentiality-preserving refinement *of the system A iff there exists a retrieve relation R mapping the data of C to the data of A with inverse R^{-1} such that:*

1. *Q_c is a behavioral refinement of Q_a, i.e. $Q_a \sqsubseteq_R Q_c$, and*
2. $$\forall r, s : traces(Q_a);\ t : traces(Q_c) \bullet r \equiv_a s \Rightarrow P_c(u \equiv_c t \mid u \in R^{-1}(r)) = P_c(v \equiv_c t \mid v \in R^{-1}(s))$$

Even though this definition mentions traces only, refusals are taken into account implicitly: after performing a certain trace s, the sum of the probabilities of all possible further behaviors of the system, i.e., events and refusals, is equal to 1. If the probabilities of all possible next events do not change, the probabilities of refusals do not change either.

The definition also covers *active* adversaries, because no assumption is made on the distribution of the probabilistic non-determinism of the abstract system. Therefore, active adversaries, imposing a particular distribution on the events at the system interface, cannot distinguish more behavior than Condition 2 allows them to, i.e., they cannot gain information from their knowledge of that distribution.

Even adversaries not adhering to the interface protocol prescribed by the abstract system are covered, because the condition of correct behavioral refinement implies that the concrete system either is tolerant against protocol violating attacks or that it prevents such attacks.

Example. To prove that *ASystem* is behaviorally refined by *CSystem*, we need to find a suitable retrieve relation. The data on the channels *inp* and *out* do not change. For the data on *w*, however, the retrieve relation must translate ciphertexts to suitable lengths of messages. This means, it must relate a given concrete trace *t* to all abstract traces *s* where the events on *inp* and *out* (obtained by the functions *Inp* and *Out*) are the same as the corresponding ones on *t*, and the events on *w* are the lengths of possible decipherings of the ciphertexts in *t*.

$$
\begin{aligned}
R = \{ & t : traces(CSystem);\ s : traces(ASystem) \mid \\
& \#t = \#s \wedge \\
& Inp(t) = Inp(s) \wedge Out(t) = Out(s) \wedge \\
& (\forall i : \operatorname{dom} t;\ ct \bullet t(i) = w.ct \\
& \quad \Rightarrow s(i) = w.(length(decipher(ct, Key(t,i)))))\}
\end{aligned}
$$

The term $decipher(ct, Key(t,i))$ is equal to $Msg(t,i)$, but we prefer to use the former in the definition of R to make the relation between corresponding items in t and s explicit.

Proving $ASystem \sqsubseteq_R CSystem$ is straightforward, and we do not go into details here.

It remains to show that the refinement $ASystem \sqsubseteq_R CSystem$ preserves confidentiality. The inverse of R is defined as follows:

$$
\begin{aligned}
R^{-1} = \{ & s : traces(ASystem);\ t : traces(CSystem) \mid \\
& \#s = \#t \wedge \\
& Inp(t) = Inp(s) \wedge Out(t) = Out(s) \wedge \\
& (\forall i : \operatorname{dom} s;\ n \bullet s(i) = w.n \\
& \quad \Rightarrow t(i) = w.(cipher(Msg(s,i), Key(t,i))))\}
\end{aligned}
$$

With that definition of R^{-1}, we can instantiate Condition 2 of Definition 3.

$$
\begin{aligned}
& \forall r, s : traces(ASystem);\ t : traces(CSystem) \bullet \\
& \quad \#r = \#s \wedge \\
& \quad (\forall i : \operatorname{dom} s \bullet \\
& \qquad length(Msg(s,i)) = length(Msg(t,i))) \\
& \qquad \Rightarrow P_c(Indist(u,t) \mid Repr(u,r)) = \\
& \qquad\qquad P_c(Indist(v,t) \mid Repr(v,s))
\end{aligned}
\tag{1}
$$

Here, the indistinguishability $Indist(u,t)$ of u (or v) from t, and the condition $Repr(u,r)$ that u is a representation of r (or v of s) are given by

$$
\begin{aligned}
& Indist(u,t) \Leftrightarrow \#u = \#t \wedge \\
& \quad \forall i : \operatorname{dom} u \bullet cipher(Msg(u,i), Key(u,i)) = \\
& \qquad cipher(Msg(t,i), Key(t,i))
\end{aligned}
$$

$$
\begin{aligned}
& Repr(u,r) \Leftrightarrow \#u = \#r \wedge \\
& \quad Inp(u) = Inp(r) \wedge Out(u) = Out(r) \wedge \\
& \quad (\forall i : \operatorname{dom} r;\ n \bullet r(i) = w.n \\
& \qquad \Rightarrow u(i) = w.cipher(Msg(r,i), Key(u,i)))
\end{aligned}
$$

$Indist(u,t)$ requires that the lengths of u and t are equal. By $Repr(u,r)$ and $Repr(v,s)$ and the assumption $\#r = \#s$, the lengths of u and v are equal. Therefore, if $\#r \neq \#t$, then both probabilities in (1) are equal to 0.

In the following, we assume that the lengths of all involved traces are equal. This entails that the domains of u and v are equal, too. From the definition of *CSystem*, we know that for all $i \neq j$ the keys $Key(u,i)$ and $Key(u,j)$ are chosen independently, and that the same holds for the keys in v. Therefore, it suffices to consider corresponding data transmissions in the involved traces independently. This means, instead of probabilistically choosing u according to distribution P_c, we choose a key $k_{u,i}$ according to distribution P. For all $i \in \operatorname{dom} t$, the required equality between probabilities is therefore equivalent to

$$
\begin{aligned}
& P(cipher(Msg(r,i), k_{u,i}) = cipher(Msg(t,i), Key(t,i)) \\
& \quad \mid cipher(Msg(r,i), k_{u,i}) = cipher(Msg(r,i), k_{u,i})) \\
& = \\
& P(cipher(Msg(s,i), k_{v,i}) = cipher(Msg(t,i), Key(t,i)) \\
& \quad \mid cipher(Msg(s,i), k_{v,i}) = cipher(Msg(s,i), k_{v,i}))
\end{aligned}
$$

We observe that the conditions on the chosen keys $k_{u,i}$ and $k_{v,i}$ are trivially true.

The ciphertexts in t are arbitrary if only they are members of the range of *cipher*. Further, the assumptions impose a restriction on the lengths of the $Msg(u,i)$ and $Msg(v,i)$ only. Therefore, condition (1) is equivalent to

$$
\begin{aligned}
& \forall msg_u, msg_v, ct \bullet \\
& \quad length(msg_u) = length(msg_v) \wedge \\
& \quad ct \in \operatorname{ran} cipher \\
& \quad \Rightarrow P(cipher(msg_u, k_u) = ct) = \\
& \qquad P(cipher(msg_v, k_v) = ct)
\end{aligned}
\tag{2}
$$

This condition is not universally true. If we assume, however, that *cipher* is defined in such a way that for given *msg* and *ct* there is exactly one *k* such that $cipher(msg, k) = ct$, then we get the usual condition that all keys are chosen with equal probability.

6.3 Transitivity

In order to be useful at all, refinement must be transitive. This means that several consecutive refinement steps can be performed, and each of the concrete system specifications is a refinement of the original abstract system specification.

Theorem 1 (Transitivity)
Let $A = (Q_a, w)$, $B = (Q_b, w)$, and $C = (Q_c, w)$ be system specifications where $A \sqsubseteq_{R_{ba}} B$ and $B \sqsubseteq_{R_{cb}} C$, and both refinements preserve confidentiality. Then $A \sqsubseteq_{R_{ba} \fatsemi R_{cb}} C$, and this refinement preserves confidentiality.

Here, $R_{ba} \fatsemi R_{cb}$ denotes the forward relational composition of R_{ba} and R_{cb}.

Proof

CSP refinement is known to be transitive. We therefore concentrate on the question whether the refinement $A \sqsubseteq_{R_{ba} \fatsemi R_{cb}} C$ preserves confidentiality.

We assume that the refinements $A \sqsubseteq_{R_{ba}} B$ and $B \sqsubseteq_{R_{cb}} C$ preserve confidentiality, i.e. we know:

$$\begin{aligned}&\forall\, r_a, s_a : traces(Q_a);\ t_b : traces(Q_b) \bullet\\ &\quad r_a \equiv_a s_a\\ &\quad \Rightarrow P_b(u_b \equiv_b t_b \mid u_b \in R_{ba}^{-1}(r_a)) =\\ &\qquad\quad P_b(v_b \equiv_b t_b \mid v_b \in R_{ba}^{-1}(s_a))\end{aligned} \tag{3}$$

$$\begin{aligned}&\forall\, r_b, s_b : traces(Q_b);\ t_c : traces(Q_c) \bullet\\ &\quad r_b \equiv_b s_b\\ &\quad \Rightarrow P_c(u_c \equiv_c t_c \mid u_c \in R_{cb}^{-1}(r_b)) =\\ &\qquad\quad P_c(v_c \equiv_c t_c \mid v_c \in R_{cb}^{-1}(s_b))\end{aligned} \tag{4}$$

Now we must show:

$$\begin{aligned}&\forall\, r_a, s_a : traces(Q_a);\ t_c : traces(Q_c) \bullet\\ &\quad r_a \equiv_a s_a\\ &\quad \Rightarrow P_c(u_c \equiv_c t_c \mid u_c \in (R_{cb} \fatsemi R_{ba})^{-1}(r_a)) =\\ &\qquad\quad P_c(v_c \equiv_c t_c \mid v_c \in (R_{cb} \fatsemi R_{ba})^{-1}(s_a))\end{aligned} \tag{5}$$

We first observe that the involved systems A, B, and C are stochastically independent as far as their probabilistic behavior is concerned $(*)$, because the probabilistic nondeterminism of a system as defined in Section 3.1 is independent of the choices in another system.

We further note that the probability to choose a trace $u_b \in R_{ba}^{-1}(r_a)$ that is indistinguishable from t_b is equal to the total of the probabilities to choose an $u_b \in R_{ba}^{-1}(r_a)$ that is equal to a given x with $x \equiv_b t_b$.

$$\begin{aligned}&P_b(u_b \equiv_b t_b \mid u_b \in R_{ba}^{-1}(r_a)) =\\ &\quad \sum_{x \equiv_b t_b} P_b(u_b = x \mid u_b \in R_{ba}^{-1}(r_a))\end{aligned} \tag{6}$$

Now, we calculate:

$$\begin{aligned}
& P_c(u_c \equiv_c t_c \mid u_c \in (R_{cb} \fatsemi R_{ba})^{-1}(r_a)) && (7)\\
= \;& P_c(u_c \equiv_c t_c \mid u_c \in (R_{ba}^{-1} \fatsemi R_{cb}^{-1})(r_a)) && (8)\\
= \;& P_c(u_c \equiv_c t_c \mid u_c \in R_{cb}^{-1}(R_{ba}^{-1}(r_a))) && (9)\\
\overset{(*)}{=} \;& \sum_{t_b \in traces(Q_b)} \Big(P_b(u_b = t_b \mid u_b \in R_{ba}^{-1}(r_a)) \cdot && (10)\\
& \qquad P_c(u_c \equiv_c t_c \mid u_c \in R_{cb}^{-1}(t_b))\Big)
\end{aligned}$$

Indistinguishability induces a partition on traces.

$$\begin{aligned}
= \;& \sum_{[u_b] \in (traces(Q_b)/\equiv_b)} && (11)\\
& \quad \sum_{t_b \equiv_b u_b} \Big(P_b(x = t_b \mid x \in R_{ba}^{-1}(r_a)) \cdot\\
& \qquad P_c(u_c \equiv_c t_c \mid u_c \in R_{cb}^{-1}(t_b))\Big)
\end{aligned}$$

By (4) and $t_b \equiv_b u_b$,
substitute u_b for t_b and factor out constant term.

$$\begin{aligned}
\overset{(4)}{=} \;& \sum_{[u_b] \in (traces(Q_b)/\equiv_b)} \Big(P_c(u_c \equiv_c t_c \mid u_c \in R_{cb}^{-1}(u_b)) \cdot && (12)\\
& \qquad \sum_{t_b \equiv_b u_b} P_b(x = t_b \mid x \in R_{ba}^{-1}(r_a))\Big)\\
\overset{(6)}{=} \;& \sum_{[u_b] \in (traces(Q_b)/\equiv_b)} P_c(u_c \equiv_c t_c \mid u_c \in R_{cb}^{-1}(u_b)) \cdot && (13)\\
& \qquad P_b(x \equiv_b u_b \mid x \in R_{ba}^{-1}(r_a))
\end{aligned}$$

Substitute s_a for r_a.

$$\begin{aligned}
\overset{(3)}{=} \;& \sum_{[v_b] \in (traces(Q_b)/\equiv_b)} P_c(v_c \equiv_c t_c \mid v_c \in R_{cb}^{-1}(v_b)) \cdot && (14)\\
& \qquad P_b(x \equiv_b v_b \mid x \in R_{ba}^{-1}(s_a))
\end{aligned}$$

Transformations similar to (13)-(7).

$$= \; P_c(v_c \equiv_c t_c \mid v_c \in (R_{cb} \fatsemi R_{ba})^{-1}(s_a)) \tag{15}$$

7 Conclusions and Future Work

We have defined a notion of confidentiality-preserving refinement that allows developers of systems where confidentiality is an issue to proceed by stepwise refinement. Confidentiality is defined as indistinguishability of system traces, given the view of a window only. For each system, not only its behavior but also the data that need *not* be kept secret have to be specified. In refining the system, the security of solutions has to be characterized on the level of more concrete system descriptions. The refinement preserves the required confidentiality of a system if the refined system does not reveal more information – in the sense of Shannon [14] – on the abstract data than permitted by the abstract specification.

Our way of specifying confidentiality properties is robust against human error: omissions in window specifications would lead to a system that keeps more information confidential than necessary. Failing to implement such a specification may reveal such an error. It is, however, impossible that an omission in a specification leads to an insecure system.

Moreover, we explicitly take into account the fact that when refining an abstract system specification, it is inevitable that more data become distinguishable. It cannot be avoided that an adversary can gain *more* information. However, it can be prevented that confidential information can be inferred.

Our central contribution to the formal specification of confidentiality and to confidentiality-preserving refinement is that we fully take probabilistic choices of systems into account. Only considering probabilities as used in cryptographic approaches makes it possible to identify a sufficient and necessary condition for confidentiality-preserving refinement.

It is obvious that CSP as a specification language cannot express all aspects of a system that are relevant for confidentiality. We view our use of CSP in this paper as an illustration of our concepts. We are confident that our definitions are applicable to other formalisms as well, even though the techniques to verify confidentiality-preserving refinement, e.g., with respect to real-time aspects or sharing of computational resources, may differ significantly from the ones used to verify CSP refinements.

Using a toy example, we have demonstrated that our approach is feasible in principle. It remains to use our notion of refinement on more realistic examples. For this purpose, tool support will be desirable if not necessary, because manipulating large formulae by hand is tedious and error-prone. A CSP model checker such as FDR[2] and theorem proving support such as Tej and Wolff's [16] embedding of CSP in Isabelle/HOL [12] offer themselves as starting points of investigations on tool-support. Extensions of such tools supporting probabilistic reasoning are necessary.

We have proven that our confidentiality-preserving refinement is transitive. Another important property that is indispensable for the usefulness of refinement in practical applications is *compositionality*. It states that in a composite system, parts of the system may be refined in isolation, and the composite system containing the refined parts can be guaranteed to be a refinement of the original system. Although we are confident that our confidentiality-preserving refinement is compositional under certain conditions, this still must be demonstrated.

[2]see `http://www.formal.demon.co.uk/FDR2.html`

Our notion of refinement is confidentiality-preserving. However, as noted in the introduction, there is much more to security than confidentiality. We seek to extend our approach and define a formal framework for the development of *secure* systems in the broader sense of the word. For example, availability should be covered in the future.

Being formal, our approach is idealistic in the sense that it guarantees confidentiality for perfect implementations only. A worthwhile goal is to study how confidentiality can be achieved in the presence of implementation errors.

All in all, we consider the results presented in this paper as a promising starting point for a comprehensive formal treatment of security issues that works for realistic cases.

Acknowledgments. We thank Heiko Mantel for comments on a draft of this paper. We also thank the anonymous referees for pointing us to Mantel's and Toussaint's work.

References

[1] J. Allen. A comparison of non-interference and non-deducibility using CSP. In *Proceedings of the 1991 IEEE Computer Security Workshop*, pages 43–54. IEEE Computer Society Press, 1991.

[2] Canadian System Security Centre. The Canadian trusted computer product evaluation criteria (version 3.0e). Communications Security Establishment; Government of Canada, 1993.

[3] European Communities – Commission. *ITSEC: Information Technology Security Evaluation Criteria (Provisional Harmonised Criteria, Version 1.2, 28 June 1991).* Office for Official Publications of the European Communities, Luxembourg, 1991. ISBN 92-826-3004-8.

[4] J. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.

[5] J. Graham-Cumming. Laws of non-interference in CSP. *Journal of Computer Security*, 2:37–52, 1993.

[6] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[7] J. Jacob. On the derivation of secure components. In *IEEE Symposium on Security and Privacy*, pages 242–247. IEEE Press, 1989.

[8] J. Jacob. Basic theorems about security. *Journal of Computer Security*, 1:385–411, 1992.

[9] J. Jürjens. Secrecy-preserving refinement. In J. N. Oliveira and P. Zave, editors, *FME 2001: Formal Methods for Increasing Software Productivity*, LNCS 2021, pages 135–152. Springer-Verlag, 2001.

[10] J.-C. Laprie, editor. *Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese*. Springer-Verlag, 1992.

[11] H. Mantel. Preserving information flow properties under refinement. In *IEEE Symposium on Security and Privacy*. IEEE Press, 2001. to appear.

[12] L. C. Paulson. *Isabelle – A Generic Theorem Prover.* LNCS 828. Springer-Verlag, 1994.

[13] A. W. Roscoe, J. C. P. Woodcock, and L. Wulf. Non-interference through determinism. In D. Gollmann, editor, *European Symposium on Research in Computer Security (ESORICS)*, LNCS 875, pages 33–53. Springer-Verlag, 1994.

[14] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[15] J. M. Spivey. *The Z Notation – A Reference Manual*. Prentice Hall, 2nd edition, 1992.

[16] H. Tej and B. Wolff. A corrected failure-divergence model for CSP in Isabelle/HOL. In J. Fitzgerald, C. B. Jones, and P. Lucas, editors, *FME'97: Industrial Applications and Strengthened Foundations of Formal Methods*, LNCS 1313, pages 318–337. Springer-Verlag, 1997.

[17] M.-J. Toussaint. Formal verification of probabilistic properties in cryptographic protocols (extended abstract). In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *Advances in Cryptology—ASIACRYPT '91*, LNCS 739, pages 412–426. Springer-Verlag, 1991.

[18] M.-J. Toussaint. Deriving the complete knowledge of participants in cryptographic protocols. In J. Feigenbaum, editor, *Advances in Cryptology (CRYPTO '91)*, LNCS 576, pages 24–43. Springer-Verlag, 1992.

[19] M.-J. Toussaint. Separating the specification and implementation phases in cryptology. In Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, editors, *European Symposium on Research in Computer Security (ESORICS '92)*, LNCS 648, pages 77–102. Springer-Verlag, 1992.

[20] V. L. Voydock and S. T. Kent. Security mechanisms in high-level network protocols. *ACM Computing Surveys*, 15(2):135–171, 1983.

[21] G. Wolf and A. Pfitzmann. Properties of protection goals and their integration into a user interface. *Computer Networks*, 32:685–699, 2000.