Threat Scenarios as a Means to Formally Develop Secure Systems[⋆]

Volkmar Lotz

Siemens AG, Corporate Research and Development, ZFE T SN 3
D–81730 München
e–mail: Volkmar.Lotz@zfe.siemens.de

Abstract. We introduce a new method for the formal development of secure systems that closely corresponds to the way secure systems are developed in practice. It is based on Focus, a general-purpose approach to the design and verification of distributed, interactive systems. Our method utilizes threat scenarios which are the result of threat identification and risk analysis and model those attacks that are of importance to the system's security. We describe the adversary's behaviour and influence on interaction. Given a suitable system specification, threat scenarios can be derived systematically from that specification. Security is defined as a particular relation on threat scenarios and systems. We show the usefulness of our approach by developing an authentic server component, thereby analysing two simple authentication protocols.
Keywords. Security, Formal Methods, Threat Identification, Risk Analysis, Stream Processing Functions, Authentication, Protocols.

1 Introduction

When developing IT-systems for security critical applications, it is of particular importance to show that the proposed solution maintains security. Formal methods can be used to prove security on a mathematically sound basis, provided an appropriate formalization of security is given. However, there is no general notion of security: for each application, different aspects of security, as confidentiality, authenticity/integrity or availability, may be relevant. Though abstract security policies may be defined, the concrete security requirements are heavily influenced by the kind of attacks that are expected for the given system and the application domain.

Informal approaches that have been shown useful in practice are therefore based on threat identification and risk analysis, where the system and its environment are investigated in detail in order to determine the kind of possible attacks, their probability, and the loss in case of the attack being performed. Thus, critical system components are identified, for which the associated risk cannot be tolerated, leading to application specific security requirements. [HMS93] gives

[⋆] This work has been carried out at Technische Universität München as part of the author's PhD research.

an overview of typical requirements on several security application domains. As in general systems are not secure by themselves, mechanisms, as for example access control, encryption or authentication protocols ([FFKK93]), have to be implemented to ensure security. It is the the system designer's task to show that a specific set of mechanisms is suitable to meet the security requirements.

A formal method for the development of secure systems that is intended to be supportive in practice should be based on the above considerations. In particular, it should employ a definition of security that is independent of security mechanisms and is therefore suitable to show the effectiveness of a mechanism. It should allow the formalization of individual security notions. Additionally, such a method should offer the opportunity of integrating security analysis and functional system development. This can be achieved by using a general-purpose method for system design and verification.

Methods achieving all these goals are currently not available. Though a lot of formal security models have been proposed and continuously developed during the last 20 years ([BLP73], [GoMe82], [TeWi89], to mention but a few), they, in general, consider specific security policies and concentrate on particular security aspects or even mechanisms. Additionally, the relationship to system implementations is often vague (one of the few exceptions is given by [BLP76]), which may explain, why these models have not been heavily used in commercial practice. Approaches ([BAN89], [Mea94], [Sne95]) dedicated to the formal analysis of certain classes of hard-to-understand mechanisms, namely authentication protocols, are promising, but often employ specification techniques and/or semantics exclusively dedicated to support security analysis and, by their nature, are not suited for the analysis of different security aspects.

In the context of process algebras, CSP in particular, there are approaches going beyond security modelling. Jacob ([Jac90]) uses inference functions describing properties of system traces in order to specify non-interference requirements. Roscoe et.al. ([RWW94]) propose to express non-interference properties as determinism of a certain system abstraction with respect to security classes and user models. In both approaches, security becomes a property of the system itself and corresponence checks with explicit security models are avoided. However, non-interference conditions seem to be too restrictive for certain applications of communication systems, for example, if only authenticity is required. Additionally, since security analysis typically occurs at early development steps, we would like to allow underspecification.

In this paper, we introduce a new formal method for the development of secure systems that is intended to meet all of the requirements mentioned above. Since we are mainly interested in applications of communication systems, we utilize a general-purpose approach to the design and verification of distributed, interactive systems. Focus ([BDD+93], [Br95], [BrSt96]) models agents by stream processing functions and is compositional with respect to refinement. In our approach, threat analysis results in the definition of threat scenarios. They are specified in Focus and can be easily derived from a system specification. Security analysis is then performed by checking the relationship between threat scenario and system specification. If the security relation holds, the threat scenario can be dropped, and system development proceeds as usual. Because of compositionality, further system refinements are secure with respect to the initial threat scenario.

Section 2 gives a brief overview of the FOCUS method and its basic notions. The properties of the semantic model of FOCUS are exploited in Sect. 3 to define threat scenarios and several notions of security that correspond to different seurity aspects. Using transmission media and typical attacks on them as example, we demonstrate how parameterized threat scenario templates can be defined. The usefulness of our approach is shown by example in Sect. 4, where we analyse two simple protocols based on ISO 9798-2 and ISO 10181-2 with respect to authenticity. The broad spectrum of our approach is indicated by the fact that the analysis of one of the protocols shows that authenticity is achieved at the expense of losing availability if an attack occurs.

2 System Specification and Development with FOCUS

In the following, we give a short introduction to the basic notions of FOCUS. We define the concepts and notations that are used in the remainder of the paper. For further reading we refer to [BDD+93] and [Br95]. The reader is expected to be familiar with set theory. We use N to denote the set of natural numbers, and $\mathsf{B} = \{0, 1\}$ to denote the set of bits. $\mathcal{P}(M)$ denotes the powerset of a set M.

2.1 Streams

In FOCUS, systems are viewed as communicating asynchronuously via named channels. Communication histories of channels are modelled by *streams* of messages, where a stream is defined to be a finite or infinite sequence of messages. Given a set of messages M, we define $M^{\overline{\omega}}$, M^{*} and $M^{\overline{\infty}}$ to denote the set of streams, finite streams and infinite streams of messages from M, respectively. We have $M^{\overline{\omega}} = M^{*} \cup M^{\overline{\infty}}$.

Streams can be viewed as functions mapping natural numbers to messages. For example, a finite stream $s \in M^{*}$ of length $n \in \mathsf{N}$ is an element of the function space $[1..n] \rightarrow M$. With $\mathsf{dom}.s$ and $\mathsf{rng}.s$ we denote the domain and the range, respectively, of a function modelling a stream.

We use $\langle\rangle$ to denote the empty stream, which is the unique finite stream that contains no messages, and $\langle m_1, m_2, \ldots, m_n\rangle$ to denote the finite stream containing the n messages m_1, m_2, $\ldots$, m_n. We utilize a number of operations on streams:

- $s \frown t$ denotes the concatenation of two streams s and t. $s \frown t$ yields the stream that starts with s and proceeds with the elements of t, if s is finite. If $s \in M^{\overline{\infty}}$, we have $s \frown t = s$. We overload the concatenation operator to messages, with $m \frown s$ denoting the result of appending the message m to s.
- $\#s$ denotes the length of a stream s with $\#s = \infty$ if $s \in M^{\overline{\infty}}$ and $\#s = n$ if $s = \langle m_1, \ldots, m_n\rangle$.

- $A©s$ denotes the stream generated from s by filtering away all elements not in A.
- $s.i$ denotes the i-th element of a stream s, if $i \leq \#s$.
- $s \sqsubseteq t$ denotes the prefix relation on streams. We have $s \sqsubseteq t$ if and only if $\exists\, r \in M^{\overline{\omega}} : s \frown r = t$.
- $s|_i$ denotes the prefix of length i of a stream s, if $i \leq \#s$, otherwise it yields s.
- $\mathsf{map}(s, f)$ for a stream s and a function f yields the stream resulting from applying f to all elements of s.
- s^n denotes the n-time iteration of the stream s. We have $s^0 = \langle\rangle$ and $s^{n+1} = s \frown s^n$.

Some of the above operators are overloaded to tuples of streams in a straightforward way. In particular, $\#(s_1, \ldots, s_n)$ yields the length of the shortest stream in $(s_1, \ldots, s_n)$, and $A©(s_1, \ldots, s_n)$ filters each stream of $(s_1, \ldots, s_n)$ with respect to A. We use the operator $(A_1 \times \ldots \times A_n)©(s_1, \ldots, s_n)$ to denote the substream of those $(s_1.i, \ldots, s_n.i)$ that are elements of $A_1 \times \ldots \times A_n$. To select the i-th element of a tuple, we use the projection function Π_i.

2.2 Timed Streams

To model the progress of time, we use so-called *timed streams*. In timed streams, the special symbol $\surd$ ("tick"), which is not an element of M, occurs. Each occurrence of $\surd$ denotes that a time unit of a particular length has passed. Messages occurring between two successive ticks are assumed to be communicated within the same time unit. Since time never halts, each infinite timed stream contains infinitely many ocurrences of $\surd$. By M^{ω}, M^* and M^{∞} we denote the set of timed streams, finite timed streams and infinite timed streams of messages of M, respectively. We have $M^{\omega} = M^* \cup M^{\infty}$.

For timed streams, we may use all of the operators defined on (untimed) streams, with ticks interpreted as ordinary messages. Moreover, we use $s{\downarrow_j}$ to define the least prefix of S that contains j occurrences of $\surd$. $s{\downarrow_j}$ therefore describes the history of a channel up to the j-th time unit. Abstraction from time is denoted by $\bar{s}$, where $\bar{s}$ results from s by removing all ocurrences of $\surd$.

2.3 Stream Processing Functions

Focus models deterministic system components by stream processing functions. In order to distinguish channels, stream processing functions usually work on named stream tuples instead of simple stream tuples. We define named stream tuples by assigning names to the input and output channels of a component, and define a mapping $\alpha \in Q \to M^{\omega}$, provided a set of channel identifiers Q is given. The operators on stream tuples that have been introduced so far are overloaded to named stream tuples, if necessary. In particular, time abstraction is lifted to named stream tuples, and denoted by $\bar{\alpha}$ for a named stream tuple α.

If $Q \cap P = \emptyset$, we define $\alpha \uplus \beta$ to denote the element of $Q \cup P \to M^\omega$ such that $c \in Q \Rightarrow (\alpha \uplus \beta)(c) = \alpha(c)$ and $c \in P \Rightarrow (\alpha \uplus \beta)(c) = \beta(c)$.

Moreover, we use $\mathbf{Q}$ as a shorthand for $Q \to M^\omega$. In Sects. 3 and 4 we often identify streams and channel names, if this is expected not to cause confusion.

We model a deterministic component C with input channels I and output channels O by a function $\tau \in \mathbf{I} \to \mathbf{O}$ that maps communication histories for the input channels to communication histories for the output channels.

To correctly reflect the behaviour of real-life components, we require for each stream-processing function modelling a component, that its output at any time j is completely determined by its input received so far, which means up to time j. If additionally a possible delay of the component is considered, requiring the output at time $j+1$ being completely determined by the input up to time j, we call the function *strongly pulse driven.* The requirements on strongly pulse driven functions τ are formally described by

$$\alpha{\downarrow_j} = \beta{\downarrow_j} \Rightarrow \tau(\alpha){\downarrow_{j+1}} = \tau(\beta){\downarrow_{j+1}} \ .$$

The arrow $\overset{s}{\to}$ is used to model domains of strongly pulse driven functions.

2.4 Composition

Strongly pulse driven functions can be composed using a number of different composition operators. For the outline of our approach, we need sequential composition, parallel composition, and feedback, which are depicted in Fig. 1 below.

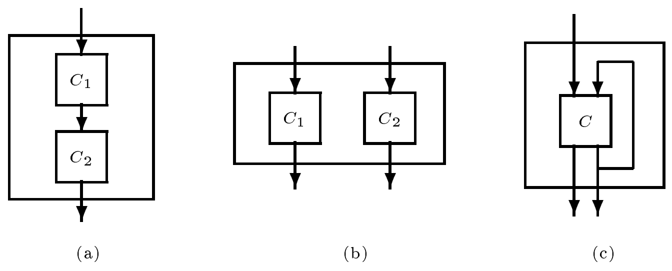

Fig. 1. Composition: (a) sequential, (b) parallel, (c) feedback

Given two strongly pulse driven stream processing functions $\tau_1 \in \mathbf{I}_1 \overset{s}{\to} \mathbf{O}_1, \tau_2 \in \mathbf{I}_2 \overset{s}{\to} \mathbf{O}_2$, we use the operator " $\succ$ " to denote sequential composition, if $O_1 = I_2$, and the operator "$\|$" to denote parallel composition, if $I_1 \cap I_2 = O_1 \cap O_2 = \emptyset$. Formally, we have

$$(\tau_1 \succ \tau_2)(\alpha) \overset{\mathsf{def}}{=} \tau_2(\tau_1(\alpha)) \ ,$$
$$(\tau_1 \parallel \tau_2)(\alpha) \overset{\mathsf{def}}{=} \tau_1(\alpha|_{I_1}) \uplus \tau_2(\alpha|_{I_2}) \ .$$

where $\alpha|_Y$ denotes the restriction of the named stream tuple α to those channels contained in Y. The functions resulting from sequential and parallel composition of strongly pulse driven stream-processing functions are strongly pulse driven as well ([BrSt96]).

Given $\tau \in \mathbf{I} \overset{s}{\to} \mathbf{O}$ we define feedback by identifying a subset of τ's output channels with a subset of τ's input channels. Let $X \subset O$ and $r \in X \to I$ be a bijection that associates a subset of τ's input channels with X. We then define $\mu_X(\tau) \in (\mathbf{I} \setminus \mathbf{r(X)}) \to \mathbf{O}$ by

$$\mu_X(\tau)(\alpha) = \beta \qquad \textsf{where} \quad \beta = \tau(\alpha \uplus \beta|_{r(X)}) \ .$$

Because of the properties of strongly pulse driven stream-processing functions, it can be shown that for each α there is a unique β that satisfies the above equation. Moreover, $\mu_X(\tau)$ is itself strongly pulse driven ([BrSt96]).

Network components are modelled by sets of stream processing functions, with this set being a singleton, if the component is deterministic. For a component $C \subseteq \mathbf{I} \overset{s}{\to} \mathbf{O}$ we define the set $C_{i/o}$ of input/output-behaviours by

$$C_{i/o} = \{(\alpha, \beta) \mid \exists \tau \in C : \tau(\alpha) = \beta\}.$$

The composition operators for stream processing functions are lifted uniformly to components. If C, C_1 and C_2 are appropriately defined, we have

$$\begin{aligned} C_1 \succ C_2 &= \{\tau \in \mathbf{I} \overset{s}{\to} \mathbf{O} \mid \forall \alpha : \exists \tau_1 \in C_1, \tau_2 \in C_2 : \tau(\alpha) = (\tau_1 \succ \tau_2)(\alpha)\} \ , \\ C_1 \parallel C_2 &= \{\tau_1 \parallel \tau_2 \mid \tau_1 \in C_1 \wedge \tau_2 \in C_2\} \ , \\ \mu_X(C) &= \{\tau \mid \forall \alpha \in (\mathbf{I} \setminus \mathbf{r(X)}) : \exists \tau' \in C : \tau(\alpha) = \mu_X.\tau'(\alpha)\} \ . \end{aligned}$$

The specific kind of the definitions for sequential composition and feedback is provided in order to achieve full abstractness of the semantic model, see [Br95] and [BrSt96] for details.

2.5 Specifications

FOCUS provides many different specification formats, whose semantics are based on the mathematical model introduced above. For our purposes, we are particularly interested in time-independent (ti) and time-dependent (td) specifications. Let I be a set of input channel names and O be a set of output channel names. The two specification formats are syntactically given by

$$\begin{aligned} S &\equiv (I \rhd O) \overset{\textsf{ti}}{::} R \ , \\ S &\equiv (I \rhd O) \overset{\textsf{td}}{::} R \ , \end{aligned}$$

where S is the name of the specification, and R is a predicate logic formula with elements of I and O as its only free variables. Semantically, a specification is interpreted to describe the set of strongly pulse driven stream processing functions that “satisfy” R.

To formally define the semantics of a specification, we first define what it means for a named stream tuple to satisfy a predicate: For any named stream tuple $\alpha \in C \to M^{\infty}$, and formula P, whose free variables are contained in C, we define $\alpha \models P$ to hold iff P evaluates to true when each free variable c in P is interpreted as $\alpha(c)$. We then define the denotation of the time-independent and time-dependent specification format by

$$\llbracket\ S\ \rrbracket \overset{\mathsf{def}}{=} \{\tau \in \mathbf{I} \overset{s}{\to} \mathbf{O} \mid \forall \alpha : \overline{(\alpha \uplus \tau(\alpha))} \models R\}\ ,$$
$$\llbracket\ S\ \rrbracket \overset{\mathsf{def}}{=} \{\tau \in \mathbf{I} \overset{s}{\to} \mathbf{O} \mid \forall \alpha : (\alpha \uplus \tau(\alpha)) \models R\}\ ,$$

respectively. Note the use of the time abstraction operator for named stream tuples in the first line.

Specifications can be composed using the same composition operators as defined for components. Since specifications describe components, the semantics of composite specifications is straightforward. Composite specifications can be syntactically given in an operator style, using the composition operators, or in a constraint style, using equations on named channels and renaming. Due to its better readability, the constraint style is often preferred in practice.

2.6 Refinement

When formally developing systems, the notion of refinement plays a central role. Focus offers a number of refinement techniques ([Br93]), of which only behaviour refinement is of interest for the following exposition. With respect to behaviour refinement, a system defined by a specification T is said to refine a system given by a specification S, if each function modelling a behaviour of T also describes a behaviour of S. If T refines S, we write $S \rightsquigarrow T$ and formally define

$$S \rightsquigarrow T \equiv \llbracket\ T\ \rrbracket \subseteq \llbracket\ S\ \rrbracket\ .$$

In order to prove that T is a refinement of S, it suffices to show that $R_T \Rightarrow R_S$.

3 System Security

3.1 Development of Secure Systems

In practice, the development of secure systems is performed in a stepwise manner. We start with a system specification S_0 which describes a system that, in general, is not secure. S_0 has to be modified by introducing security mechanisms which counter those threats that have been identified as critical. The system S_1 resulting from this modification should be a refinement of S_0, since suitable security mechanisms are expected not to affect the specified system behaviour. Constructing a secure system is an iterative process, since security mechanisms introduce new components and/or data to the system which may themselves be subject to attack and have to be secured by further mechanisms. For example, considering a cryptographic mechanism that relies on secret keys, we need a mechanism to keep these keys confidential. For each iteration, we identify the following activities to be carried out (let S_i be the starting system specification, which will be refined to S_{i+1}):

1. **Threat Identification and Risk Analysis.** This is an application specific task that has to be carried out each time a security analysis is to be performed. Though classes of possible threats can be defined with respect to component types and application domains, the actual assessment of threats and associated risks heavily depends on the given system S_i. For example, if transmission media are considered, the associated risk depends, among others, on whether they are located in a secure or in a public area. Threat identification and risk analysis results in a classification of system components with respect to their criticality, and a description of the attacks that critical components may be subject to. Threat descriptions are concrete in the sense of referring to particular system components, and multiple occurences of the same kind of threat are possible (for example, if there are several communication links that are assumed to be eavesdropped).
2. **Definition of Threat Scenario.** The results of threat identification and risk analysis are used to specify a formal threat scenario B_i, in which critical components are replaced by subsystems that specify the relevant attacks. Thus, B_i models the system behaviour in a situation where all of the relevant attacks occur, which is the worst case with respect to security. Obviously, B_i is not necessarily a refinement of S_i.
3. **Selection or Development of Mechanisms.** During this activity, suitable security mechanisms are selected or developed, where "suitable" means that the mechanisms are able to counter the threats as well as that they satisfy further criteria, including non-technical ones as, for example, cost and performance.
4. **Refinement.** S_i is extended by a specification of the selected mechanisms. We yield a system specification S_{i+1} and, implicitly, a refined threat scenario B_{i+1}. It has to be shown that S_{i+1} is a refinement of S_i.
5. **Security Analysis.** In order to justify the selection of mechanisms in step 3, we have to show that S_{i+1} is secure, which is performed by proving that the security property holds with respect to S_{i+1} and B_{i+1}. The concrete structure of the security property depends on the security policy and the security requirements, see Sect. 3.3 for details. If the proof fails, the selected mechanisms are not appropriate, and steps 3 to 5 have to be repeated.

Starting from S_0, we yield a sequence of system specifications S_1, S_2, $\ldots$, S_n. The process is finished with a secure system S_n, if the risk analysis does not identify further threats that have to be countered, or the remaining threats are countered by non-technical mechanisms that are beyond the scope of our approach. Thus, step 1 must always follow step 5, which ensures that new threats resulting from the introduction of mechanisms are always considered. However, it often turns out to be useful to already include such new threats in the construction of B_{i+1}, which, for example, is done in Sect. 4.

Our approach aims at the formal foundation of the development steps described above. However, risk analysis and selection of mechanisms are excluded, since they heavily depend on non-technical arguments and thus are out of reach of formal treatment. Since all of the formal work is performed within the FOCUS framework, at each time of security development there is a unique relationship to system development according to its functional specification. However, methodological issues of integrated functional and security development are beyond the scope of this paper, and further work will be dedicated to this subject.

3.2 Threat Scenarios

A threat scenario is a modification of a system specification that describes a situation in which the system is attacked by an adversary, according to the results of threat identification and risk analysis. In most application cases, the threat scenario can be derived systematically from the system specification: threat identification and risk analysis are typically performed on the basis of an architectural view of the system, which means that we have a compositional specification as starting point of security considerations. For each of the components, it can then be determined, whether it is likely to be subject to adversary actions. In the derivation of a threat scenario, the critical components will then be replaced by specifications modelling the adversary's influence on them.

Candidates for critical components can often be defined on the basis of an analysis of the application domain and the type of the component, or its role within the system specification. This offers the opportunity of defining templates describing abstract attacks on the component types of interest. Using instantiations of these templates for the modification of critical components identified by risk analysis, application specific threat scenarios can be easily constructed. Note that not necessarily each of the components of a given type has to be replaced, but if risk analysis leads to a specific component of that type being classified as critical, the template can be used.

In distributed communication systems and networks, it is mainly the communication medium rather than the communicating entities (users or computer systems) that are considered to be at risk (imagine logical communication channels being implemented by using public telephone lines). Therefore, in order to perform a risk analysis reasonably, we require the specification to explicitly model media as network agents, using an appropriate level of abstraction. However, this does not seem to cause problems in practice: if the medium is subject to further development, for example if it is going to be implemented by a protocol working on an unreliable physical medium, it will be explicitly specified, otherwise it can be simply modelled by an agent behaving like the identity on its input. In the following, we provide a template for the construction of threat scenarios describing attacks on communication media. Given the results of the threat identification and risk analysis for a particular link of the system to be secured, the template can be easily instantiated, leading to an appropriate threat scenario for the given link. This will be demonstrated in Sect. 4.

Suppose M being a set of arbitrary messages, and MD being the specification of a medium transmitting messages of M, formally defined by

$$\mathrm{MD} \equiv (i : M \triangleright o : M) :: \quad R_{MD} \ ,$$

with R_{MD} being an arbitrary predicate describing the communication behaviour of MD. If risk analysis identifies MD as critical, in the worst case an adversary is able to eavesdrop communication as well as to influence the transmission behaviour of the channel. Such an attack can be modelled by a network as depicted in Fig. 2, which replaces MD in the threat scenario construction. The

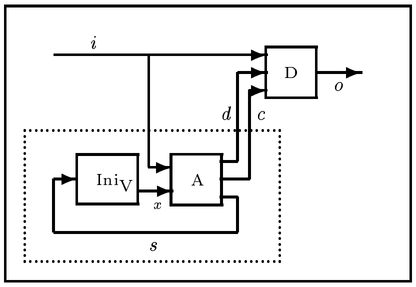

Fig. 2. Threat scenario for communication channels

threat scenario template is based on an explicit model of the adversary, together with the initial information available to her and the set of functions she can use to compute new information. As in [Sne95] and [Mea94], we use an explicit model of the adversary's influence on communication, based on the semantic model of Focus: the "data flow component" D specifies how the adversary influences the behaviour of the transmission medium. For example, the adversary may insert or delete messages. Obviously, the specification of D has to take into account properties of the medium MD, leading to D being a function of MD. A formal specification of the threat scenario $\mathrm{MD}_{\mathrm{Thr}}$, an instance of which is to replace each specification of a critical medium of the system analysed is given below. For better readability, the specification is given in constraint style.

$$\begin{aligned} &\mathrm{MD}_{\mathrm{Thr}} \equiv (i : M \triangleright o : M) :: \\ &\quad (o) := \mathrm{D}(\mathrm{MD})(i, d, c), \quad (d, c, s) := \mathrm{A}(i, x), \quad (x) := \mathrm{Ini}_V(s) \ . \end{aligned}$$

The specification is basically divided into two parts: D models the influence on communication, and the subnetwork consisting of Ini_V and A (indicated by the dotted box in Fig. 2) describes the adversary's abilities to generate new messages. Let U be a set of values, elements of which the adversary may use to perform her attacks. $V \subseteq U$ represents the set of values that are initially available to the adversary. Each time the adversary eavesdrops a message sent by a client, this set of values is extended according to the contents of this message and the set of functions the adversary may use to compute new values from already known ones. Let $F \subseteq \left(\bigcup_{n \in \mathsf{N}} U^n \to U\right) \times \mathsf{N}$ be a set of functions together with their arities that operate on messages, formally, if $n \in \mathsf{N}$ and $(f, n) \in F$, then $f \in U^n \to U$. The set of new messages C_F the adversary may get by stepwise computation from V using functions from F is then given by

$$C_F(V) = \bigcup_{n \in \mathsf{N}} C_F^N(n, V) \ ,$$

where $C_F^N(0, V) = V$ and $C_F^N(m+1, V) = \{x \in U \mid \exists (f, n) \in F, x_1, \ldots, x_n \in C_F^N(m, V) : x = f(x_1, \ldots, x_n)\}$.

Note that we are only interested in values satisfying the type constraints on MD's interface, since other values do not help the adversary in compromising the system. The formal specification of the components relating to the adversary is given by

$$\mathrm{Ini}_V \equiv (s : \mathcal{P}(U) \rhd x : \mathcal{P}(U)) \overset{\mathsf{ti}}{::} x = V \frown s \ ,$$

$$\begin{aligned}
&\mathrm{A} \equiv (i : M, x : \mathcal{P}(U) \rhd d : M, c : C, s : \mathcal{P}(U)) \overset{\mathsf{ti}}{::} \\
&\quad \#d = \#x \quad \wedge \\
&\quad \forall j \in \mathsf{dom}.x : \\
&\qquad d_j \in (x_j \cup \{i_j\}) \quad \wedge \\
&\qquad j \in \mathsf{dom}.i \Rightarrow s_j = C_F(x_j \cup \{i_j\})\}) \quad \wedge \\
&\qquad j \notin \mathsf{dom}.i \Rightarrow s_j = x_j \ .
\end{aligned}$$

At each point j, x_j contains the set of messages known to the adversary. Whenever she is able to eavesdrop a message from i, the set of messages will be updated according to the functions in F. The adversary may use each of the messages available to influence communication, e.g. by inserting them. Such fraudulent messages issued by the adversary are modeled by d. In some applications, it may turn out to be necessary to explicitly specify the influence of the adversary on the legitimate entity's communication, for example by determining the point of time at which a fraudulent message is inserted. We use c to model this kind of control, where data from a set of controls C are issued. Typically, we have $C = \mathsf{B}$. Within the template, we do not impose further restrictions on c, however, in an instantiation of the template further constraints can be introduced.

Note that, from a technical point of view, the specification could as well have been given by modelling the set of values available to the adversary as an internal state of A, thus saving the component Ini_V. However, we chose the above specification in order to explicate the central role of the adversary's knowledge in the context of security analysis.

In our template for attacks on communication channels, the data flow component D is not further specified, since the adversary's influence on communication is considered to be application specific. However, the syntactical interface of D (legitimate messages on i, fraudulent messages on d, and controls on c) allows all kinds of possible attacks, as for example listed in [Mun93], to be specified. Often, reliability aspects of the medium and specific attack descriptions can be separated, leading to a simple structure of D with respect to its parameter MD:

$$\mathrm{D}(\mathrm{MD}) \equiv (i, d, c \rhd o) :: \quad \mathrm{D}' \succ \mathrm{MD}$$

for some D'. If, for example, the adversary may only insert new messages, without infinitely blocking legitimate messages, but is not able to determine the position where to insert, D' is given by the specification of the fair merge agent in [BDD+93].

This concludes the specification of the threat scenario template for transmission media. Its parameters are given by the adversary's initial set of values V, the set F of functions available, the type of control messages C, and the specification of the data flow component D. In addition, for some applications it may be suitable to further strengthen A. Section 4 shows a sample use of this template.

The kind of adversary model used in the threat scenario specification is close to the approach taken in [Sne95] and [Mea94], where it turned out to be useful for the analysis of cryptographic protocols. Differences occur, however, in the explicit modelling of the adversary's influence on communication, which in our approach can be tailored to the application at hand.

3.3 The Security Property

Given a system specification S and a threat scenario B that has been derived from S, security can be expressed using a particular binary relation R_{Sec} on specifications. If $R_{Sec}(B, S)$ holds, S is said to be secure with respect to the threats represented in B. However, the implications of $R_{Sec}(B, S)$ on the security of a system being implemented according to S depend heavily on the concrete definition of R_{Sec}. In the remainder of this section we want to introduce a number of variants of such a definition, which correspond to different kinds of security notions. Thus, our interpretation of security is split into two parts: a system specific part, which relates to vulnerabilities of the system under development, the specific abilities of an attacker to that system, and the environment of it, being modelled in a threat scenario, and a general part expressing common security requirements, being modelled using a particular security relation.

We start with the definition of the most restrictive type of security, in which adversary interference is expected to have no influence on the behaviour of the system. In this case, the threat scenario must be a refinement of the original system.

Definition 1. A system S with syntactic interface (I,O) is called *absolutely secure* with respect to a threat scenario B, with the same interface, if $R_S(\mathrm{B},\mathrm{S})$ holds, with R_S being defined by $R_S(\mathrm{B},\mathrm{S}) \equiv \mathrm{S} \rightsquigarrow \mathrm{B}$. $\square$

In practice, absolute security is usually hard to achieve, and sometimes it is even not desired: if there are interactions that are not considered to be security relevant, then an adversary may influence these without compromising security.

If the security requirements on the application at hand are known exactly, we may use only these to define the system's security.

Definition 2. Given a predicate P, a system S with syntactic interface (I,O) is called *P-secure* with respect to a threat scenario B, with the same interface, if $P(\mathrm{B})$ holds. $\square$

For certain common aspects of security, like integrity, authenticity, confidentiality, or availability, formal definitions can be provided. Using these definitions in a security analysis, the analyst need not formalise particular security requirements, but may only use the definition covering the aspects that are of importance to her application. Since in Sect. 4 we focus on authentication mechanisms, we provide a general definition for authenticity of a system. Similar definitions can be given for the other aspects mentioned.

Definition 3. A system S with syntactic Interface (I,O) is called *authentic* with respect to a threat scenario B, with the same interface, if $R_{Ath}(\mathrm{B},\mathrm{S})$ holds, with R_{Ath} being defined by

$$\begin{aligned} R_{Ath}(\mathrm{B},\mathrm{S}) \equiv{}& \forall f \in \mathbf{I} \overset{s}{\to} \mathbf{O} : \\ & f \in [\![\ \mathrm{B}\ ]\!] \Rightarrow \forall x \in \mathbf{I} \quad \exists h \in \mathsf{B}^{\overline{\omega}}, f' \in [\![\ \mathrm{S}\ ]\!] : f'(\mathrm{sel}(x,h)) = f(x) \ , \end{aligned}$$

where $\forall x \in \mathbf{I}, h \in \mathsf{B}^{\overline{\omega}} : \ \mathrm{sel}(x,h) = \Pi_1((M \times 1) ⓒ (x,h))$. $\square$

The above definition states, that, if $(x, f(x))$ is an i/o-behaviour of B, then there is a substream x' of x such that $(x', f(x))$ is an i/o-behaviour of S, with the appropriate substream being selected by an oracle h and the function sel. This means that each output of B is caused by a "legitimate" input, but we do not require the attacked system to respond to all legitimate inputs.

3.4 Security Mechanisms

When threat identification and risk analysis is performed, systems, in general, turn out not to be secure. Therefore, we have to specify particular means, called security mechanisms, that are suited to counter the threats that have been identified as critical. We distinguish between technical mechanisms, which are given by a particular functionality of an IT system, and non-technical mechanisms, which are organisational or physical means located in the system's environment. As an example of non-technical means, take a messenger delivering a secret key, or a door lock preventing an intruder from accessing a computer system located in a particular room. In our approach, we only consider technical mechanisms, since they form a part of the system to be developed and can therefore be treated in the same way as functional requirements. However, assumptions based on non-technical mechanisms may influence the adversary model.

A lot of basic technical mechanisms suited to meet different security requirements have been proposed. [FFKK93] gives a representative overview. In general, for a given security problem, there are several mechanisms that are suited to meet the requirements, differing only with respect to non-functional criteria as performance, cost, and legal issues (patents, licences). Though these criteria may be of major importance to the application, they do not contribute to security analysis as described in the previous sections. Therefore, the selection problem is considered to be out of scope of our approach.

The mechanisms we are particularly interested in, include those based on cryptographic methods. They are based on concepts as common secrets, cryptographic keys, random numbers, nonces, and so on. In our approach, each of these concepts is modelled by a specific data type, where the adversary's abilities on the usage of elements of these data types are restricted. Consider, for example, the set of cryptographic keys and cryptograms in Sect. 4. The model of communication and the semantics of FOCUS allows to benefit from results of approaches specifically dedicated to the description of cryptographic systems, for example [Sne95] or [Mea94].

4 A Sample Development

4.1 A Simple Server

Our example provides the specification of a very simple, idealized server component that is able to receive requests submitted by a client via a transmission medium and to respond to those requests that have been issued by authorized clients by sending results using a different communication channel. Since the main focus of the example is on security analysis of the server, the detailed structure and contents of requests and results are not important. However, if looking for possible applications for servers of this kind, imagine a door lock which is only released upon request, for example by inserting a smart card, or a mobile phone system, in which connect requests are received by a server and, possibly supplied with additional data about the requestor, forwarded to a switching center. We assume that there are several clients using the same request channel, thus each request has to be tagged with the client's identifier. Figure 3 shows an abstract view of the server, consisting of a server component SV and the transmission medium MD. To formally specify the server in FOCUS, let Id be a set of identifiers, $L \subseteq \mathit{Id}$ the set containing the identifiers of the authorized clients and Req, Res represent the set of requests and results, respectively. As

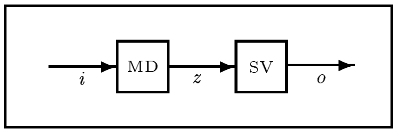

Fig. 3. A simple server

argued above, *Req* and *Res* are not specified in detail. Using the operator style of specification, the server is described by

$$\mathrm{S} \equiv (i : \mathit{Id}{\times}\mathit{Req} \rhd o : \mathit{Res}) :: \quad \mathrm{MD} \succ \mathrm{SV} \ ,$$

with the component specifications given by

$$\mathrm{MD} \equiv (i : \mathit{Id}{\times}\mathit{Req} \rhd z : \mathit{Id}{\times}\mathit{Req}) \overset{\mathsf{ti}}{::} \quad z = i \ ,$$

$$\mathrm{SV} \equiv (z : \mathit{Id}{\times}\mathit{Req} \rhd o : \mathit{Res}) \overset{\mathsf{ti}}{::} \quad \#o = \#(L \copyright \mathsf{map}(z, \Pi_1)) \ .$$

Note that we assume an ideal transmission medium, resulting in the component MD being simply the identity on its input channel. This has been chosen in order to keep the simplicity of the example. Section 3.2 outlines how one may deal with more sophisticated media specifications.

SV states that each request of an authorized client, and only those, will be served. Because of the semantic model of time independent specifications in Focus, SV ist quite implicit: from the strong guardedness constraint on functions satisfying SV it follows that requests are served in order of their receipt, and that no responses are issued in advance, anticipating future requests.

4.2 The Threat Scenario

In Sect. 3.2 we stated that each threat scenario is the result of an application specific threat identification and risk analysis, where templates can be used in the construction of the scenario. Since risk analysis heavily depends on non-technical arguments, for example consideration of associated financial loss, it is not completely covered by our method. For our example, we therefore assume that a risk analysis has been carried out, with the supposed result of the adversary being assessed as being able to eavesdrop the transmitted messages, to know about the set of possible and legitimate clients and requests, and to insert fraudulent messages. Additionally, we assume that the adversary inserts at most one fraudulent message between two legitimate messages (note that this requirement is only added to keep the example simple). These assumptions completely describe the adversary's behaviour, particularly she cannot manipulate or delete messages on i in our example scenario.

Since MD models a transmission medium as discussed in Sect. 3.2, the template given there can be used to construct the threat scenario B. Thus,

$$\mathrm{B} \equiv (i : \mathit{Id}{\times}\mathit{Req} \rhd o : \mathit{Res}) :: \quad \mathrm{MD}_{\mathrm{Thr}} \succ \mathrm{SV} \ ,$$

with $\mathrm{MD}_{\mathrm{Thr}}$ as defined in Sect. 3.2, using the message set $M = \mathit{Id}{\times}\mathit{Req}$. Let $V = M$ and $F = \emptyset$, which states that the adversary knows the complete set of messages that may be transmitted. Moreover, let $C = \mathsf{B}$ be the set of control messages. The assumptions on the adversary are modelled by adding the following conjunct to the specification of A in the scenario template $\mathrm{MD}_{\mathrm{Thr}}$ of Sect. 3.2:

$$\#1 \copyright c = \#d \quad \wedge \quad \exists i, j \in \mathsf{N}, k \in \mathsf{N} \cup \{\infty\} : \quad c = (\langle 0 \rangle^i \frown \langle 1 \rangle \frown \langle 0 \rangle^j)^k \ .$$

The first conjunct states that for each message in d we have a corresponding 1 in c, and the second that c contains no substring s with more than one 1's in a row.

We still have to instantiate the data flow component D of $\mathrm{MD}_{\mathrm{Thr}}$. Since we have decided to strengthen the adversary model A, we may use a quite general specification of D, which will be suited for other analyses as well. We define

$$\begin{aligned} &\mathrm{D} \equiv (i : M, d : M, c : \mathsf{B} \rhd z : M) \overset{\mathsf{ti}}{::} \\ &\quad i \sqsupseteq \Pi_1((M{\times}0) \copyright (z, c' \frown 0^\infty)) \quad \wedge \quad d|_n = \Pi_1((M{\times}1) \copyright (z, c')) \\ &\quad \textsf{where} \quad n = \min(\#d, \#1 \copyright c) \quad \wedge \quad c' \sqsubseteq c \quad \wedge \quad \#1 \copyright c' = n \ . \end{aligned}$$

Note that D to some extent corresponds to the specification of a merge component, with the oracle partly determined by the control sequence c. Fairness of D depends on the control sequence input: if, and only if, the control sequence allows the insertion of infinitely many messages, transmission of messages of i may be suspended for ever, this fact being reflected by using the prefix relation instead of equality with respect to i, and by extending the control sequence in the first conjunct. On the other hand, given an appropriate control sequence, each of the adversary's messages will indeed be inserted. Potential loss of fairness is intentional, since it does not seem to be reasonable to always assume a fair adversary. The auxiliary values n and c' are introduced to handle cases where the control sequence and the messages sent by the adversary do not fit together, meaning that there are less 1's in c than messages in d or vice versa. However, from our specification of A, we always have appropriate control sequences.

S is not authentic with respect to B, as is shown in the following theorem.

Theorem 4. S *is not authentic w.r.t.* B*, i.e.* $R_{Ath}(\mathrm{B}, \mathrm{S})$ *does not hold.*

Proof. Choose $i = \langle\rangle$, $d = \langle(id_0, rq)\rangle$ for some $id_0 \in L$, $rq \in \mathit{Req}$, and $c = \langle 1 \rangle$ as existential witnesses. Then, $(i, (d, c))$ is a possible i/o-behaviour of $\mu(\mathrm{Ini}_V \succ \mathrm{A})$. In this case, by the definition of D, we have $z = \langle(id_0, rq)\rangle$, leading to $\#o = 1$

by the definition of SV. Since for all $h \in \mathsf{B}^{\overline{\omega}}$, $\mathrm{sel}(h, \langle\rangle) = \langle\rangle$, authenticity of S would require (i, o) to be an i/o-behaviour of S, which is obviously not the case, because for all $f \in [\![S]\!]$, we have $\#x = \#f(x)$. □

4.3 Two Authentication Protocols

In order to specify an authentic server, we have to refine S by introducing an appropriate security mechanism. ISO proposes two variants of a simple challenge-response authentication protocol ([ISO92], [ISO93]) that are considered to be suited for applications like our server. The protocols are based on symmetric cryptoalgorithms and random number generators, and assume that the server and each of the legitimate clients share a secret key. To model cryptographic systems, a value space as for example defined in [Sne95] is suited for our stream based communication model as well.

To describe the cryptographic system used in our example, let K be a set of cryptographic keys, Cr a set of cryptograms, and Ms a set of messages. We have an encryption function $E : K \times (Cr \cup Ms) \to Cr$ and a decryption function $D : K \times (Cr \cup Ms) \to (Cr \cup Ms)$. In symmetric cryptosystems, we have

$$\begin{array}{ll} D(k, E(k, x)) = x, & x \in Ms \cup Cr \ , \\ E(k, x_1) = E(k, x_2) \Rightarrow x_1 = x_2, & k \in K, x_1, x_2 \in Ms \cup Cr. \end{array}$$

Further properties hold with high probability. Since Focus, like almost all other approaches to distributed systems design and verification, is not intended to deal with probabilities, we have to approximate them by predicate logic formulæ. A reasonable idealization is to take properties that hold with high probability for granted.

It is considered to be improbable that the adversary constructs cryptograms (by simply guessing or taking arbitrary keys and messages – which in good cryptosystems both are of nearly equal probability) that match cryptograms being issued by legitimate users. We model this fact by

$$\begin{array}{ll} E(k_1, x) = E(k_2, x) \Rightarrow k_1 = k_2, & k_1, k_2 \in K, x \in Ms \cup Cr, \\ E(k_1, m_1) = E(k_2, m_2) \Rightarrow k_1 = k_2 \wedge m_1 = m_2, & k_1, k_2 \in K, m_1, m_2 \in Ms, \end{array}$$

and assume that the adversary does not exploit the finiteness of the set of cryptograms. Note that the latter formula is modelled to only hold for messages of Ms, and requires that Ms is of considerably less cardinality than Cr.

The protocols also use random numbers. We choose a set R of values from which random numbers are taken. For each stream $r \in R^{\overline{\omega}}$ of random numbers, we at least require that no duplications occur, described by $PRN(r)$, with

$$PRN(r) \equiv \forall j \in \mathsf{dom}.r : r.j \notin \mathsf{rng}.\,(r|_{j-1}) \ .$$

PRN obviously does not completely characterize random numbers, but is sufficient to show authenticity of one of the protocols specified below. We are

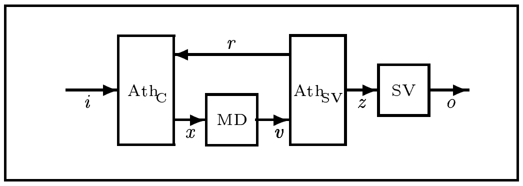

Fig. 4. An authentication mechanism

now ready to specify the authentication protocols. They both share the same structure, as depicted in Fig. 4. Each time a request is received by Ath_{SV}, it issues a challenge on r and proceeds only in the case that the next message received is an appropriate response to the challenge. Otherwise, the request will be ignored. Ath_{C} is responsible for passing on requests and suitable responses, if challenges are received. The two protocols differ only in the type of challenges (ISO 9798-2: random numbers, ISO 10181-2: encrypted random numbers) and responses (ISO 9798-2: encrypted random numbers, ISO 10181-2: random numbers). For simplicity of the example, we specify a slight abstraction of the ISO-protocols, namely without considering the inclusion of SV's identifier in the response. Thus, we lose protection against reflection attacks, which is, however, of less importance with respect to the demonstration of how our approach works. The refined server is specified by

$$\text{SA}_i \equiv (i : \mathit{Id}{\times}\mathit{Req} \rhd o : \mathit{Res}) :: \quad \mu\,(\text{Ath}^i_{\text{C}} \succ \text{MD} \succ \text{Ath}^i_{\text{SV}}) \succ \text{SV} \ ,$$

with $i \in \{1, 2\}$ indicating the actual variant of the protocol. MD and SV are specified as in Sect. 4.1. In the component specifications, we do not only describe a single run of the protocol, but also specify how the authentication agents serve multiple requests. For the moment, we assume that Ath_{C} buffers all incoming challenges.

Each legitimate client shares a secret key with the server, meaning that there is a set $K_0 \subseteq K$ with $K_0 = \{k_{id} | id \in L\}$. For $i = 1$ denoting the variant based on ISO-9798-2, we then have (with $M = \mathit{Id}{\times}\mathit{Req}$ as in Sect. 4.2)

$$\begin{aligned}
&\text{Ath}^1_{\text{C}} \equiv (i : M, r : R \rhd x : M \cup \mathit{Cr}) \overset{\text{ti}}{::} \\
&\exists\ f_1 \in (M^{\overline{\omega}} \times R^{\overline{\omega}}) \to (M \cup \mathit{Cr})^{\overline{\omega}}, f_2 \in (\mathit{Id} \times M^{\overline{\omega}} \times R^{\overline{\omega}}) \to (M \cup \mathit{Cr})^{\overline{\omega}} : \\
&\qquad x = f_1(i, r)
\end{aligned}$$

where $\forall i \in M^{\overline{\omega}}, r \in R^{\overline{\omega}}, (id, req) \in M, rn \in R :$

$$
\begin{aligned}
&f_1(\langle\rangle, r) = \langle\rangle, \\
&id \in L \Rightarrow f_1((id, req) \frown i, r) = (id, req) \frown f_2(id, i, r), \\
&id \notin L \Rightarrow f_1((id, req) \frown i, r) = f_1(i, r), \\
&\\
&f_2(id, i, rn \frown r) = E(k_{id}, rn) \frown f_1(i, r) \ \ .
\end{aligned}
$$

and

$$\mathrm{Ath}^1_{\mathrm{SV}} \equiv (v : M \cup Cr \rhd r : R, z : M) \overset{\mathsf{ti}}{::}$$

$$
\begin{aligned}
&\exists\ f_3 \in (M \cup Cr)^{\overline{\omega}} \to (R^{\overline{\omega}} \times M^{\overline{\omega}}), \\
&\qquad f_4 \in (Id \times Req \times R \times (M \cup Cr)^{\overline{\omega}}) \to (R^{\overline{\omega}} \times M^{\overline{\omega}}) : \\
&\qquad z = f_3(v, r) \quad \wedge \quad PRN(r)
\end{aligned}
$$

where $\forall i \in (M \cup Cr)^{\overline{\omega}}, (id, req) \in M, rn \in R, cr \in M \cup Cr :$

$$
\begin{aligned}
&f_3(\langle\rangle) = (\langle\rangle, \langle\rangle), \\
&id \in L \Rightarrow \exists rn \in R : f_3((id, req) \frown v) = (\langle\rangle, rn) \frown f_4(id, req, rn, v), \\
&\\
&D(k_{id}, cr) = rn \Rightarrow f_4(id, req, rn, cr \frown v) = ((id, req), \langle\rangle) \frown f_3(v), \\
&\neg D(k_{id}, cr) = rn \Rightarrow f_4(id, req, rn, cr \frown v) = f_3(v) \ \ .
\end{aligned}
$$

The specification is given in a rather operational style, which corresponds to the way cryptographic protocols are typically presented. Both are underdetermined, which in our case does not matter because of the particular composition of SA_i. If any of the agents is waiting for a response, anything except the response awaited will be rejected, with the agent set back in a state where it waits for new requests.

The specification for the second variant, based on ISO 10181-2, is entirely similar, with the type of challenges and responses exchanged, and the type of channels adjusted.

In order to show that SA_i describes an appropriate security mechanism, we first have to prove that SA_i is a refinement of S.

Theorem 5. *For $i \in \{1, 2\}$, SA_i is a behaviour refinement of S, i.e. for all $f \in M^{\overline{\omega}} \to Req^{\overline{\omega}}$ we have $f \in [\![\ \mathrm{SA}_i\ ]\!] \Rightarrow f \in [\![\ \mathrm{S}\ ]\!]$.*

Proof. We show the theorem only for $i = 1$. The proof for the second variant of the protocol is entirely similar. Since MD describes the identity, with respect to the specification of SV it suffices to show, for all $i \in M^{\overline{\omega}}, f_{\mathrm{Ath_C}} \in [\![\ \mathrm{Ath_C}\ ]\!], f_{\mathrm{Ath_{SV}}} \in [\![\ \mathrm{Ath_{SV}}\ ]\!]$, that

$$\mu\,(f_{\mathrm{Ath_C}} \succ f_{\mathrm{Ath_{SV}}})(i) = L \copyright \mathsf{map}(i, \Pi_1)$$

holds. This is shown by proving, if $(z,r) = (f_{\mathrm{Ath_C}} \succ f_{\mathrm{Ath_{SV}}})(i,r)$ for a fix point r, then $z = L \copyright \mathsf{map}(i, \mathit{\Pi}_1)$. Using induction on i, this is shown for finite i, and by continuity of the functions involved, for infinite i. The equation above then follows from the definition of feedback.

$i = \langle\rangle$. We have $f_{\mathrm{Ath_{SV}}}(f_{\mathrm{Ath_C}}(\langle\rangle, r)) = (\langle\rangle, r)$.
$i = (id_0, req_0) \frown i'$.
If $id_o \notin L$, we have $(z,r) = f_{\mathrm{Ath_{SV}}}(f_{\mathrm{Ath_C}}(i,r)) = f_{\mathrm{Ath_{SV}}}(f_{\mathrm{Ath_C}}(i',r))$, with the assertion following immediately from the induction hypothesis.
If $id_0 \in L$, we have

$$\begin{aligned}(z,r) &= f_{\mathrm{Ath_{SV}}}(f_{\mathrm{Ath_C}}(i,r)) \\ &= f_{\mathrm{Ath_{SV}}}((id_0, req_0) \frown f_2(id_0, i', r)) \\ &= (\langle\rangle, rn) \frown f_4(id_0, req_0, rn, f_2(id_0, i', r))\end{aligned}$$

for some $rn \in R$. From the fix point property of r, we conclude $r = rn \frown r'$ and further yield

$$\begin{aligned}(z,r) &= (\langle\rangle, rn) \frown f_4(id_0, req_0, rn, E(k_{id_0}, rn) \frown f_{\mathrm{Ath_C}}(i', r')) \\ &= (\langle\rangle, rn) \frown ((id_0, req_0), \langle\rangle) \frown f_{\mathrm{Ath_{SV}}}(f_{\mathrm{Ath_C}}(i', r')) \ .\end{aligned}$$

We therefore have r' being a fix point with respect to i', which from the induction hypothesis and $id_0 \in L$ leads to the assertion. □

Now, the last step remaining in developing an authentic server is to show that SA_i, $i \in \{1,2\}$ indeed satisfies the authenticity definition of Sect. 3.3. Since with the definition of the security mechanism additional channels and new message types have been introduced, it is appropriate to update the threat scenario parameters. For our example, we assume that challenges are transmitted via a secure channel (remember Fig. 4, where the threatened medium is only specified for the request and response channel), but that the adversary knows the set of possible random values R, and thus can guess one of them. In addition, she has some keys available, but not those of the legitimate clients, and may encrypt as well as decrypt. We further assume that the adversary only inserts fake requests and immediately tries to give an appropriate authentication response. The threat scenario instantiation is then given by $V = M \cup R \cup K_A$ for some $K_A \subseteq K \setminus K_0$, $F = \{E, D\}$, D as defined in Sect. 4.2, and A strenghtened by

$$\begin{aligned}&\exists\, h \in \mathsf{B}^{\overline{\omega}}, n \in \mathsf{N} \cup \{\infty\} : h = \langle 0,1\rangle^n \wedge \mathrm{sel}(h,d) \in Cr^{\overline{\omega}} \wedge \mathrm{sel}(\bar{h},d) \in M^{\overline{\omega}} \ \wedge \\ &\exists\, i,j \in \mathsf{N}, k \in \mathsf{N} \cup \{\infty\} : c = (\langle 0\rangle^{2i} \frown \langle 1,1\rangle \frown \langle 0\rangle^{2j})^k \ ,\end{aligned}$$

with $\bar{h}$ denoting the bitwise complement of a bitstream h. Note that the basis for this strengthening is the adversary specifcation A of Sect. 3.2, not the one from Sect. 4.2. With this threat scenario instantiation BA_1, we can show

Theorem 6. SA_1 *is authentic w.r.t.* BA_1*, i.e.* $R_{Ath}(\mathrm{SA}_1, \mathrm{BA}_1)$ *holds.*

Proof. It suffices to show that, if $(z,r) = f_{\mathrm{Ath}_{\mathrm{SV}}}(f_{\mathrm{MD}_{\mathrm{Thr}}}(f_{\mathrm{Ath}_{\mathrm{C}}}(i,r)))$ for $i,z \in M^{\overline{\omega}}, r \in R^{\overline{\omega}}, f_{\mathrm{Ath}_{\mathrm{C}}} \in [\![\mathrm{Ath}_{\mathrm{C}}]\!], f_{\mathrm{MD}_{\mathrm{Thr}}} \in [\![\mathrm{MD}_{\mathrm{Thr}}]\!], f_{\mathrm{Ath}_{\mathrm{SV}}} \in [\![\mathrm{Ath}_{\mathrm{SV}}]\!]$, then there is $h \in \mathsf{B}^{\overline{\omega}}$ such that

$$z = \mathrm{sel}(h, L \copyright \mathsf{map}(i, \Pi_1)) \ .$$

Let $j \in \mathsf{N}$ be the highest index with $f_{\mathrm{Ath}_{\mathrm{C}}}((L{\times}Req \copyright i)|_j, r) \sqsubseteq f_{\mathrm{MD}_{\mathrm{Thr}}}(f_{\mathrm{Ath}_{\mathrm{C}}}(i,r))$, i.e. the adversary for the first time inserts a message after the j-th legitimate request. Since we have $\#r \geq \#i$ for each fix point r, from the definition of $\mathrm{Ath}_{\mathrm{C}}$ we can conclude $f_{\mathrm{Ath}_{\mathrm{C}}}((L{\times}Req \copyright i)|_j, r) = f_{\mathrm{Ath}_{\mathrm{C}}}((L{\times}Req \copyright i)|_j, r|_j)$. Thus,

$$\begin{aligned}(z,r) &= f_{\mathrm{Ath}_{\mathrm{SV}}}(f_{\mathrm{MD}_{\mathrm{Thr}}}(f_{\mathrm{Ath}_{\mathrm{C}}}(i,r))) \\ &= ((L{\times}Req \copyright i)|_j, r|_j) \frown f_{\mathrm{Ath}_{\mathrm{SV}}}(v \frown f_{\mathrm{MD}_{\mathrm{Thr}}}(f_{\mathrm{Ath}_{\mathrm{C}}}(i',r'))) \ ,\end{aligned}$$

with $i = i|_j \frown i'$, $r = r|_j \frown r'$, and, from the definition of $\mathrm{MD}_{\mathrm{Thr}}$ which reflects the assumption on the adversary, $v = \langle (id_0, req_0), e \rangle$ for some $id_0 \in L$, $req_0 \in Req$, and $e \in Cr$. This leads to

$$(z,r) = ((L{\times}Req \copyright i)|_j, r|_j) \frown (\langle\rangle, rn) \frown f_4(id_0, req_0, rn, e \frown f_{\mathrm{MD}_{\mathrm{Thr}}}(f_{\mathrm{Ath}_{\mathrm{C}}}(i',r')))$$

for some $rn \in R$. We now either have $e = E(k, rn')$, $k \in K_A \subseteq K \setminus K_0, rn' \in R$, i.e. a cryptogram constructed by the adversary, or $e = E(k_{\Pi_1((L{\times}Req \copyright i).j')}, r.j')$ for some $j' \leq j$, i.e. a cryptogram eavesdropped by the adversary. From $PRN(r)$ it follows that $rn \notin \mathsf{rng}.r|_j$, thus it follows by the properties of the cryptographic system, that the authentication fails, leading to

$$(z,r) = ((L{\times}Req \copyright i)|_j, r|_j) \frown (\langle\rangle, rn) \frown f_{\mathrm{Ath}_{\mathrm{SV}}}(f_{\mathrm{MD}_{\mathrm{Thr}}}(f_{\mathrm{Ath}_{\mathrm{C}}}(i',r'))) \ .$$

By an inductive argument we can the show that each following authentication fails as well (since $\mathrm{Ath}_{\mathrm{C}}$ always takes an “old” challenge), leading to

$$(z,r) = ((L{\times}Req \copyright i)|_j, r) \ ,$$

from which the assertion follows (with $h = \langle 1 \rangle^j \frown \langle 0 \rangle^\infty$). □

SA_2 is not authenthic, if the analoguous threat scenario instantiation BA_2 is considered.

Theorem 7. SA_2 *is not authentic w.r.t.* BA_2, *i.e.* $R_{Ath}(\mathrm{SA}_2, \mathrm{BA}_2)$ *does not hold.*

Proof. Since the set of random values is available to the adversary, she may guess an appropriate response without knowledge of the challenge cryptogram. Take $i = \langle\rangle$, $r = \langle rn \rangle$, $d = \langle (id_0, req_0), rn \rangle$, and $c = \langle 1, 1 \rangle$ as existential witnesses. □

SA_2 can only be shown to be authentic, if further restrictions on the adversary behaviour are introduced. For example we may exclude R from the set of values initially available to the adversary, and require that a new challenge cannot be predicted from the eavesdropped ones, which means that for each $r \in R, j \in \mathsf{N}$, there is no function $f \in F$ such that $r.(j+1) = f(r.1, \ldots, r.j)$.

Unfortunately, the proof of theorem 6 shows an unpleasant property of our authentication mechanism: if a fraudulent request is once issued, all forthcoming legitimate requests will be denied, since an additional challenge is issued, and due to the buffering of challenges $\mathrm{Ath}_{\mathrm{C}}$ and $\mathrm{Ath}_{\mathrm{SV}}$ will never synchronize. To avoid this effect in order to increase availability of the system, we need a time dependent specification of $\mathrm{Ath}_{\mathrm{C}}$ which ignores all challenges until it has issued a new request. Such an $\mathrm{Ath}'_{\mathrm{C}}$ can be specified as follows:

$$\mathrm{Ath}'_{\mathrm{C}} \equiv (i : M, r : R \rhd x : M \cup Cr) \overset{\mathsf{td}}{::}$$

$$\begin{aligned}&\exists f_1 \in (M^\omega \times R^\omega) \to (M \cup Cr)^\omega, f_2 \in (Id \times M^\omega \times R^\omega) \to (M \cup Cr)^\omega : \\ &\quad x = \surd \frown f_1(i,r)\end{aligned}$$

$$\text{where} \quad \forall i \in M^\omega, r \in R^\omega, (id, req) \in M, rn \in R :$$

$$\begin{aligned}
&f_1(\surd \frown i, \surd \frown r) = \surd \frown f_1(i,r),\\
&f_1(\surd \frown i, rn \frown r) = \surd \frown f_1(i,r),\\
&id \in L \Rightarrow f_1((id,req) \frown i, \surd \frown r) = \surd \frown (id,req) \frown f_2(id,i,r),\\
&id \in L \Rightarrow f_1((id,req) \frown i, rn \frown r) = (id,req) \frown f_2(id,i,r),\\
&id \notin L \Rightarrow f_1((id,req) \frown i, \surd \frown r) = \surd \frown f_1(i,r),\\
&id \notin L \Rightarrow f_1((id,req) \frown i, rn \frown r) = f_1(i,r),\\
\\
&f_2(id,i,\surd \frown r) = \surd \frown f_2(id,i,r),\\
&f_2(id,i,rn \frown r) = E(k_{id},rn) \frown f_1(i,r).
\end{aligned}$$

With the same proof techniques as employed in the proof of theorem 6, we can show that a server using $\mathrm{Ath}'_{\mathrm{C}}$ is authentic as well.

5 Conclusion and Further Work

We have introduced a new approach to the formal development of secure systems that is based on a procedure being established in practice and aims at a mechanism independent security notion, flexibility with respect to security aspects as well as integration of security analysis and development according to the functional requirements on the system. Application specific security requirements, as a result of threat identification and risk analysis, are formally modelled by threat scenarios which specify the anticipated behavior of the adversary, in particular her influence on communication. Security is defined as a relation on threat scenarios and systems.

The main focus of this paper has been to show the basic principles of our approach by conducting a comprehensive sample development of an authentic server. For purposes of presentation, our example has been simplified: we focus exclusively on authentication, provide simple protocols, and restrict the behaviour of the adversary. However, our example is of practical relevance, since the protocols are only slight abstractions of standard protocols ([ISO92], [ISO93]) and the adversary characterization seems to be reasonable for certain application situations (for example, a secure door lock).

The example shows a number of promising results that raise evidence that the approach is well-suited to support the formal development of secure systems in practice. Firstly, with respect to the asumptions on the adversary, we have been able to prove the authenticity of SA_1, and to show that SA_2 is not authentic. Moreover, the construction of the proof of authenticity of SA_1 leads to a revision of the protocol specification: though preserving authenticity, the first specification, which buffers challenges, turns out to lack availability in case of an attack, leading to a time dependent protocol specification SA'_1. Thus, our method turns out to be suitable for the analysis of effects resulting from multiple executions of protocols, because the semantic model guarantees the consideration of the whole lifetime of the system instead of just a single protocol run. Additionally, it offers the opportunity to reason about different security aspects. Formal definitions of several security notions have been given.

Applicability of our method is supported by dividing the security notion in an application specific part (threat scenario) and a general part (security relation). In common applications, threat scenarios may be derived systematically from compositional system specifications, which has been shown for components modelling transmission media in communication systems.

Our approach particularly benefits from choosing Focus as the basis of formalization. Since Focus is a general purpose formal development method, it offers the opportunity to continue system development from those specifications that result from security analysis. On the other hand, security analysis can be performed at each stage of the system development. Systematic derivation of threat scenarios is supported: information flow to the adversary is modelled by simply adding (logical) channels to the system specification.

However, a lot of work remains to be done: the approach has to be generalized by defining further security relations, corresponding, for example, to confidentiality and availability. Effects of multiple attacks, which may occur if an adversary is able to simultaneously attack several critical components, and of interleaving of protocol runs have to be investigated. To improve practicability, it is important to provide a set of threat scenario templates that can be instantiated for a variety of common threat analysis results, and a set of basic mechanism specifications. The approximation of cryptographic algorithms has to be further improved. A notion of compositionality with respect to different threats and threatened components is desirable.

Even in its initial state, our approach provides significant progress with respect to a formal method that reaches the aims mentioned above. With further work being performed, we will get close to a method that can be profitably applied in practice.

References

[BLP73] D.E. Bell, L. LaPadula: Secure Computer Systems: Mathematical Foundations (NTIS AD-770 768), A Mathematical Model (NTIS AD-771 543), A Refinement of the Mathematical Model (NTIS AD-780 528), MTR 2547 Vol. I-III, ESD-TR-73-278, Mitre Corporation, Bedford MA, 1973

[BLP76] D.E. Bell, L. LaPadula: Secure Computer Systems: Unified Exposition and Multics Interpretation, NTIS AD-A023 588, MTR 2997, ESD-TR-75-306, Mitre Corporation, Bedford MA, 1976

[BDD+93] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T.F. Gritzner, R. Weber: The Design of Distributed Systems – An Introduction to FOCUS – Revised Version, Technical Report TUM-19202-2, Technische Universität München, 1993

[Br93] M. Broy: (Inter-)Action Refinement: The Easy Way, in: M. Broy (Ed.): Program Design Calculi, NATO ASI Series F, Vol. 118, Springer, 1993

[Br95] M. Broy: Advanced Component Interface Specification, in: T. Ito, A. Yonezawa (Eds.): Theory and Practice of Parallel Programming, Proceedings TPP '94, Springer LNCS 907, 1995

[BrSt96] M. Broy, K. Stølen: Interactive System Design, Book Manuscript, 1996

[BAN89] M. Burrows, M. Abadi, R. Needham: A Logic of Authentication, Report 39, Digital Systems Research Center, Palo Alto, 1989

[FFKK93] O. Fries, A. Fritsch, V. Kessler, B. Klein (Hrsg.): Sicherheitsmechanismen: Bausteine zur Entwicklung sicherer Systeme, REMO Arbeitsberichte, Oldenbourg Verlag, München 1993 (in German)

[GoMe82] J.A. Goguen, J. Meseguer: Security Policies and Security Models, Proc. of the IEEE Symposium on Security and Privacy, 1982, pp. 11–20

[HMS93] S. Herda, S. Mund, A. Steinacker (Hrsg.): Szenarien zur Sicherheit informationstechnischer Systeme, REMO Arbeitsberichte, Oldenbourg Verlag, München 1993 (in German)

[ISO92] ISO/IEC CD 9798: Information Technology – Security Techniques – Entity Authentication Mechanisms, Part 2: Entity Authentication Using Symmetric Techniques, 1992

[ISO93] ISO/IEC DIS 10181-2.2: Information Technology – Open Systems Interconnection – Security Framework for Open Systems: Authentication Framework, 1993

[Jac90] J.L. Jacob: Specifying Security Properties, in: C.A.R. Hoare (ed.): Developments in Concurrency and Communications, Addison-Wesley, 1990

[Mea94] C. Meadows: The NRL Protocol Analyzer: An Overview, Journal of Logic Programming, Vol. 19, 1994

[Mun93] S. Mund: Sicherheitsanforderungen – Sicherheitsmaßnahmen, VIS '93 (Herausgeber: P. Horster, G. Weck), Vieweg Verlag, 1993 (in German)

[RWW94] A.W. Roscoe, J.C.P. Woodcock, L. Wulf: Non-interference through Determinism, in: D. Gollmann: Computer Security – ESORICS '94, Springer LNCS 875, 1994

[Sne95] E. Snekkenes: Formal Specification and Analysis of Cryptographic Protocols, PhD thesis, 1995

[TeWi89] P. Terry, S. Wiseman: A 'New' Security Policy Model, Proc. of the IEEE Symposium on Security and Privacy, 1989, pp. 215–228

This article was processed using the LaTeX macro package with LLNCS style