Part V

Appendices

A

Towards UML 2.0

At the time of writing, the next major version of the Unified Modeling Language, UML 2.0, is being finalized. In this chapter, we shortly sketch how to accommodate the changes from the current version, UML 1.5, with respect to the approach proposed in this book. A good introduction to UML 2.0 and a list of the changes between the different UML versions is contained in [Fow04].

Most changes do not have a significant impact on our approach: For example, *object diagrams*, which already exist in UML 1.5, are in UML 2.0 more explicitly defined. In UML 1.5, packages can be contained in class diagrams, while in UML 2.0 such diagrams are now independently named as *package diagrams*. The UML 1.5 collaboration diagrams are called *communication diagrams* in UML 2.0. The UML 2.0 *interaction overview diagrams* are a new kind of diagram integrating activity and sequence diagrams. – While in UML 1.5, the activities in activity diagrams can already be defined using other diagrams, such as sequence diagrams, this link can be made more explicitly in UML 2.0 by actually including the sequence diagrams in the respective activity states. *Timing diagrams* are a kind of diagram which is entirely new to UML with version 2.0. These diagrams, which will be familiar to many hardware engineers from electronic engineering, do not seem to be particularly specific to secure software engineering.

Composite structure diagrams have been included in UML 2.0 from the real-time UML-RT extension. They contain *parts* represented by rectangles that may be connected by *connectors* drawn as lines between parts. These diagrams are useful for specifying component structures and hierarchies within components. In that, they are similar to the static structure diagrams of UML 1.5, which can contain subsystems that may themselves in turn contain static structure diagrams, and to component models, which can be part of deployment diagrams. Thus, stereotypes such as «secrecy», «integrity», «authenticity», and «high» defined for dependencies in static structure diagrams and deployment diagrams can also be applied to the connectors in composite structure diagrams. Also, parts can be marked as «critical», such as class models or subsystems in static structure diagrams. Although the

UMLsec notation can be extended quite nicely to UML 2.0 composite structure diagrams, one should note that this information can already be expressed in UML 1.5 static structure and deployment diagrams, so using UML 1.5 is not a restriction in this respect. Note that while deployment diagrams and component diagrams are integrated in UML 1.5, they are written in different diagrams in UML 2.0.

UML 2.0 also adds quite a few new model elements for existing diagram types. These include state machine extensions, gates in interaction diagrams, and power types in class diagrams. They are not very particular to secure software engineering but are also not in conflict with the UMLsec notation and can be used within the context of the approach presented in this book without any problems. Similarly, some existing model elements have been changed, but most of these do not appear in our treatment in this book at all. In particular, we do not make use of any UML 1.5 model elements that have been dropped in UML 2.0. In UML 2.0 sequence diagrams, *interaction frames* extend the guards used in UML 1.5 sequence diagrams to specify conditional behavior. Again, this extension can be used with UMLsec as well, although it remains to be seen in which situations the added expressivity outweighs the increase of complexity in the notation. Stereotypes are in UML 2.0 more tightly defined than before and exclude a previous usage as a kind of keywords, but include the usage of stereotypes in UMLsec. Activity diagrams are defined more liberally in UML 2.0 than before: While UML 1.x views activity diagrams formally as a special case of statechart diagram, this imposes some constraints on the structure of a diagram that are removed in UML 2.0, for example that forks and joins have to match. To accommodate this liberalization, the semantics is now formulated in a Petri-net style by referring to token flows. Swimlanes can be multidimensional in UML 2.0 and are called partitions.

B

The Semantics of UML Machine Rules

We give a formal definition for the semantics of UML Machine rules. It is inspired by those for Abstract State Machines in [SSB01, BS00].

Definition B.1 (Update). *An* update *for a UML Machine A is a triple $(f, a_1, \ldots, a_n), b)$, where f is an n-ary function name, and $a_1, \ldots, a_n$ and b are elements of the base set of A.*

Thus an update specifies that the interpretation of the function f in A has to be changed at the arguments $a_1, \ldots, a_n$ to the value b. An update set is a set of updates.

For two update sets U, V, we define the update set $U; V$ (U followed by V) as follows: $U; V \stackrel{\text{def}}{=} \{(f, a, b) \in U : \neg\exists c.(f, a, c) \in V\} \cup V$.

A transition rule of a UML Machine produces, in any given state, an update set for each variable assignment. Recursive calls to other rules are allowed; thus it is possible that a rule has no well-defined semantics at all. For the calculus that defines the semantics of transition rules in Fig. B.1 we need some further technical definitions.

Given a UML Machine A, a term t over $\mathbf{Voc}\, A$, a state S of A and a variable assignment ζ which assigns the variables in t to elements of the base set X of A, we write $[\![t]\!]^S_\zeta$ for the interpretation of t over X in state S which extends ζ.

We write $\zeta[x \mapsto a]$ for the variable assignment which coincides with ζ except that it assigns the element a to the variable x. Thus:

- $\zeta[x \mapsto a](v) = a$ if $v = x$
- $\zeta[x \mapsto a](v) = \zeta(v)$ otherwise.

For a rule R and $n \geq 1$, we write R^n for the rule **seq** $R \ldots R$ **endseq** that iterates R n times.

Definition B.2 (Semantics of transition rules). *The semantics of a transition rule R of a given UML Machine A with base set X in a state S with*

$$\frac{}{[\![\textbf{skip}]\!]^S_\zeta \rhd \emptyset}$$

$$\frac{}{[\![f(t) := s]\!]^S_\zeta \rhd \{(f, a, b)\}} \quad \text{if } a = [\![t]\!]^S_\zeta \text{ and } b = [\![s]\!]^S_\zeta$$

$$\frac{[\![R_1]\!]^S_\zeta \rhd U_1 \ \ldots \ [\![R_n]\!]^S_\zeta \rhd U_n}{[\![\textbf{do-in-parallel } R_1 \ \ldots \ R_n \textbf{ enddo}]\!]^S_\zeta \rhd \bigcup_i U_i}$$

$$\frac{[\![R]\!]^S_\zeta \rhd U}{[\![\textbf{if } g \textbf{ then } R \textbf{ else } R']\!]^S_\zeta \rhd U} \quad \text{if } [\![g]\!]^S_\zeta = \mathit{true}$$

$$\frac{[\![R']\!]^S_\zeta \rhd U}{[\![\textbf{if } g \textbf{ then } R \textbf{ else } R']\!]^S_\zeta \rhd U} \quad \text{if } [\![g]\!]^S_\zeta = \mathit{false}$$

$$\frac{[\![R]\!]^S_{\zeta[x \mapsto a]} \rhd U}{[\![\textbf{choose } x \textbf{ with } g \textbf{ do } R]\!]^S_\zeta \rhd U} \quad \text{if } [\![g]\!]^S_{\zeta[x \mapsto a]} = \mathit{true}$$

$$\frac{}{[\![\textbf{choose } x \textbf{ with } g \textbf{ do } R]\!]^S_\zeta \rhd \emptyset} \quad \text{if there exists no } a \text{ with } [\![g]\!]^S_{\zeta[x \mapsto a]} = \mathit{true}$$

$$\frac{[\![R]\!]^S_{\zeta[x \mapsto a]} \rhd U_a \quad \text{for each } a \in I}{[\![\textbf{forall } x \textbf{ with } g \textbf{ do } R]\!]^S_\zeta \rhd \bigcup_{a \in I} U_a} \quad \text{if } I = \{a \in X : [\![g]\!]^S_{\zeta[x \mapsto a]} = \mathit{true}\}$$

$$\frac{[\![R_1]\!]^S_\zeta \rhd U_1 \ \ldots \ [\![R_n]\!]^S_\zeta \rhd U_n}{[\![\textbf{seq } R_1 \ \ldots \ R_n \textbf{ endseq}]\!]^S_\zeta \rhd U_1 ; \ldots ; U_n}$$

$$\frac{[\![R^n]\!]^S_\zeta \rhd U_n \quad \text{for each } n \in \mathbb{N}}{[\![\textbf{iterate}(R)]\!]^S_\zeta \rhd \lim_{n \to \infty} U_n} \quad \text{if } \exists n \geq 0 : U_n = U_{n+1}$$

Fig. B.1. The semantics of UML Machine rules

respect to a variable assignment ζ is defined as an update set U such that $[\![R]\!]^S_\zeta \rhd U$ can be derived in the calculus in Fig. B.1, if such a set exists. Otherwise, it is undefined.

Note that those rules from Sect. 7.1 whose semantics is not defined in Fig. B.1 can be defined in terms of those that are listed in Fig. B.1. Note also that there can be different update sets U such that $[\![R]\!]^S_\zeta \rhd U$ is derivable in the calculus (because of the non-determinism introduced by the **choose with do** rule).

It is possible that the update set $[\![R]\!]^S_\zeta$ contains several updates for the same function name f. Then the updates have to be consistent in the following sense, otherwise the execution stops.

Definition B.3 (Consistent update set). *An update set U is called* consistent *if it satisfies the following property:*

$$\textit{If } (f, (a_1, \ldots, a_n), b) \in U \textit{ and } (f, (a_1, \ldots, a_n), c) \in U, \textit{then } b = c.$$

Thus a consistent update set contains for each function and each argument tuple at most one value.

If an update set U is consistent, it can be fired in a given state. The result is a new state in which there may be function names the interpretations of which are changed according to U.

Definition B.4 (Firing of updates). *The result of* firing *a consistent update set U in a state S of the UML Machine A is a new state T of A satisfying the following two conditions for each function name $f \in \textbf{Voc}\, A$:*

- *If* $(f, (a_1, \ldots, a_n), b) \in U$, *then* $[\![f]\!]^T(a_1, \ldots, a_n) = b$.
- *If there is no* b *with* $(f, (a_1, \ldots, a_n), b) \in U$, *then* $[\![f]\!]^T(a_1, \ldots, a_n) = [\![f]\!]^S(a_1, \ldots, a_n)$.

Definition B.5 (Run of a UML Machine). *Let* M *be a UML Machine with vocabulary* Σ, *initial state* S, *and main rule name* R. *Let* ζ *be a variable assignment. A* run $r \in$ **Run** M *of* M *is a finite or infinite sequence* $S_0, S_1, \ldots$ *of states for* Σ *such that the following conditions are satisfied:*

- $S_0 = S$.
- *For each* $n \in \mathbb{N}$, *if* S_n *is the last element of the sequence* r *then*
  - *for any update set* U *with* $[\![R]\!]^{S_n}_{\zeta} \rhd U$, *applying* U *leaves the state* S_n *unchanged, or*
  - *there exists an inconsistent update set* U *with* $[\![R]\!]^{S_n}_{\zeta} \rhd U$.
- *For each* $n \in \mathbb{N}$, *if* S_n *is not the last element of the sequence* r, *then there exists a consistent update set* U *with* $[\![R]\!]^{S_n}_{\zeta} \rhd U$ *in* S_n *such that* S_{n+1} *is the result of firing* U, *and such that* $S_{n+1} \neq S_n$.

C

Proofs

We give here proof sketches for the statements from Chaps. 5, 7, and 8. Note that the proofs for the statements in Chap. 5 are performed with respect to the formal definitions and results in Chaps. 7 and 8, rather than the informal exposition in Sect. 3.3, and are thus deferred to the end of this chapter. Note also that it is not intended to propose manual reasoning to establish security analysis results as in Chap. 5 in the context of security engineering with UMLsec in practice. Instead, tool support for analyzing UMLsec specifications should be used as discussed in Chap. 6. Manual proofs are presented here to demonstrate that UMLsec is suitable overall to express important security properties in a way that allows detailed formal security analysis.

C.1 UML Machines

Example

Fact 7.4. *For each sequence* $(I_1, \ldots, I_n)$, $[\![Sndr]\!](I_1, \ldots, I_n)$ *consists of those sequences* $(O_1, \ldots, O_n)$ *that fulfill the following conditions, for each* $i \in \{1, \ldots, n\}$*:*

- $O_i \subseteq \{\!|\, transmit \,|\!\}$.
- $\sharp(O_1 \uplus \ldots \uplus O_i) \leq \sharp(I_1 \uplus \ldots \uplus I_{i-1}) \searrow \{\!|\, send \,|\!\}$.
- *The conditions that*

  $$\sharp(I_j \uplus \ldots \uplus I_{i-1}) - \sharp((I_j \uplus \ldots \uplus I_{i-1}) \searrow \{\!|\, send \,|\!\}) < i - j - 2 * \sharp(O_j \uplus \ldots \uplus O_{i-1})$$

  for each $j < i$ *and that* $\sharp(O_1 \uplus \ldots \uplus O_{i-1}) < \sharp(I_1 \uplus \ldots \uplus I_{i-1}) \searrow \{\!|\, send \,|\!\}$ *imply* $\sharp O_i > 0$.

Proof. To see that the above characterization of the behavior of $[\![Sndr]\!](I_1, \ldots, I_n)$ is correct, one has to convince oneself that the given conditions are necessary and sufficient for a sequence $(O_1, \ldots, O_n)$ to be contained in $[\![Sndr]\!](I_1, \ldots, I_n)$, for any sequence $(I_1, \ldots, I_n)$.

We first consider necessity. The first condition is necessary, because $[\![Sndr]\!]()$ only outputs messages *transmit*, on any input. The second condition is necessary, because the UML Machine only outputs a message for each *send* that is received. The third condition is necessary because the UML Machine will output a message at execution round i provided that, firstly, there is still a *send* message in the input queue that has not yet prompted a *transmit* output, and, secondly, we have currState $= Send$, because any other input received apart from *send* messages has already been consumed.

To consider sufficiency of the conditions, suppose we are given sequences $(O_1, \ldots, O_n)$ and $(I_1, \ldots, I_n)$ such that the three conditions are fulfilled. Then $(O_1, \ldots, O_n)$ is contained in $[\![Sndr]\!](I_1, \ldots, I_n)$, because from the two sequences we can construct an internal behavior of the UML Machine *Sndr* with the sequence C_i of contents of currState which produces the sequence of outputs $(O_1, \ldots, O_n)$ given the sequence of inputs $(I_1, \ldots, I_n)$: for each i, if $O_i = \{\!\{ transmit \}\!\}$ then $C_i = Send$, otherwise $C_i = Wait$.

C.2 Refinement

Fact 7.8. *(Delayed) $\mathcal{E}$-$(\mathbf{i}, \mathbf{o})$-refinement of UML Machines is a preorder for each set of events $\mathcal{E} \subseteq$ Events and tuples $\mathbf{i}$ and $\mathbf{o}$ of input and output names.*

Proof. We show that (delayed) $\mathcal{E}$-$(\mathbf{i}, \mathbf{o})$-refinement is reflexive for each set of events $\mathcal{E} \subseteq$ **Events** and tuples $\mathbf{i}$ and $\mathbf{o}$ of input and output names. For any UML Machine A, any set $\mathcal{E} \subseteq$ **Events**, tuples $\mathbf{i}$ and $\mathbf{o}$ of input and output names, and sequence $\mathbf{I}$ of event multi-sets, we have $[\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subseteq [\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$ and $[\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subsetneq\!\!\!\!\sim\; [\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$ since $\subseteq$ and $\subsetneq\!\!\!\!\sim$ are reflexive.

We show that (delayed) $\mathcal{E}$-$(\mathbf{i}, \mathbf{o})$-refinement is transitive for each set of events $\mathcal{E} \subseteq$ **Events** and tuples $\mathbf{i}$ and $\mathbf{o}$ of input and output names. Suppose we are given the UML Machines A, A', and A'', tuples $\mathbf{i}$ and $\mathbf{o}$ of input and output names, and a set $\mathcal{E} \subseteq$ **Events**, such that A' (delayed) $\mathcal{E}$-$(\mathbf{i}, \mathbf{o})$-refines A and A'' (delayed) $\mathcal{E}$-$(\mathbf{i}, \mathbf{o})$-refines A'. To show that A'' (delayed) $\mathcal{E}$-$(\mathbf{i}, \mathbf{o})$-refines A, suppose we are given a sequence $\mathbf{I} = I_1, \ldots, I_n$ of event multi-sets with $\bigcup_i \lfloor I_i \rfloor \subseteq \mathcal{E}$. We have to show that $[\![A'']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subseteq [\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$ and $[\![A'']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subsetneq\!\!\!\!\sim\; [\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$. By assumption, we know that we have $[\![A']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subseteq [\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$ and $[\![A'']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subseteq [\![A']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$, and $[\![A']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subsetneq\!\!\!\!\sim\; [\![A]\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$ and $[\![A'']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E} \subsetneq\!\!\!\!\sim\; [\![A']\!]_{\mathbf{i},\mathbf{o}}(\mathbf{I}) \curvearrowright \mathcal{E}$. We can conclude by transitivity of $\subseteq$ and $\subsetneq\!\!\!\!\sim$.

Fact 7.10. *If the UMS $\mathcal{A}'$ is a refinement of the UMS $\mathcal{A}$ then the UML Machine* **Exec** $\mathcal{A}'$ *is a refinement of the UML Machine* **Exec** $\mathcal{A}$.

Proof. Suppose we are given UMSs $\mathcal{A}'$ and $\mathcal{A}$ such that $\mathcal{A}'$ is a refinement of $\mathcal{A}$. We need to show that the UML Machine $\mathcal{A}'$ is a refinement of the UML Machine $\mathcal{A}$.

The link structures of $\mathcal{A}$ and $\mathcal{A}'$ are the same by definition of refinement for UMSs. It is thus sufficient to show that each sequence of contents of the family of link queues $(\mathsf{linkQu}_{\mathcal{A}'}(l))_{l \in \mathsf{Links}_{\mathcal{A}'}}$ is also a sequence of contents of the family $(\mathsf{linkQu}_{\mathcal{A}}(l))_{l \in \mathsf{Links}_{\mathcal{A}}}$. This follows from the assumption that $\mathcal{A}'$ is a refinement of $\mathcal{A}$ and from the definition of refinement of UMSs, which implies that there are bijections b and $b_{\mathcal{C}}$ as in Definition 7.9 such that for each component $\mathcal{C} \in \mathsf{Comp}_{\mathcal{A}}$ and activity $A \in \mathsf{Act}_{\mathcal{C}}^{\mathcal{A}}$, the UML Machine of $b_{\mathcal{C}}(A)$ is a $(\mathbf{i}, \mathbf{o})$-refinement of the A Machine where $\mathbf{i} = \mathsf{Att}_{\mathcal{C}}^{\mathcal{A}}$ and $\mathbf{o} = \mathbf{i} \cup \{\mathsf{finished}_A\}$.

Fact 7.11. *Refinement of UMSs is a preorder.*

Proof. We show that refinement of UMSs is reflexive. For any UMS $\mathcal{A} = (\mathsf{Comp}_{\mathcal{A}}, \mathsf{Sched}_{\mathcal{A}}, \mathsf{Links}_{\mathcal{A}}, \mathsf{Msgs}_{\mathcal{A}})$, the identity functions $b \stackrel{\text{def}}{=} \mathsf{id} : \mathsf{Comp}_{\mathcal{A}} \to \mathsf{Comp}_{\mathcal{A}}$ (and similarly the $b_{\mathcal{C}}$) fulfill the required conditions by Fact 7.8.

We show that refinement of UMSs is transitive. Suppose we are given UMSs $\mathcal{A} = (\mathsf{Comp}_{\mathcal{A}}, \mathsf{Sched}_{\mathcal{A}}, \mathsf{Links}_{\mathcal{A}}, \mathsf{Msgs}_{\mathcal{A}})$, $\mathcal{A}' = (\mathsf{Comp}_{\mathcal{A}'}, \mathsf{Sched}_{\mathcal{A}'}, \mathsf{Links}_{\mathcal{A}'}, \mathsf{Msgs}_{\mathcal{A}'})$, and $\mathcal{A}'' = (\mathsf{Comp}_{\mathcal{A}''}, \mathsf{Sched}_{\mathcal{A}''}, \mathsf{Links}_{\mathcal{A}''}, \mathsf{Msgs}_{\mathcal{A}''})$, such that $\mathcal{A}'$ *refines* $\mathcal{A}$ and $\mathcal{A}''$ *refines* $\mathcal{A}'$. Thus we have bijections $b : \mathsf{Comp}_{\mathcal{A}} \to \mathsf{Comp}_{\mathcal{A}'}$ and $b' : \mathsf{Comp}_{\mathcal{A}'} \to \mathsf{Comp}_{\mathcal{A}''}$ (and similarly for the $b_{\mathcal{C}}$) fulfilling the above conditions. To show that $\mathcal{A}''$ *refines* $\mathcal{A}$, we note that the bijection $b' \circ b : \mathsf{Comp}_{\mathcal{A}} \to \mathsf{Comp}_{\mathcal{A}''}$ (and similarly the $b_{\mathcal{C}}$) fulfills the conditions as well, by Fact 7.8.

Fact 7.12. *Suppose we are given a parameterized UMS $\mathcal{A}(\mathcal{Y}_1, \ldots, \mathcal{Y}_n)$, where the activity variable $\mathcal{Y}_i$ belongs to the component $\mathcal{C}_i$, for each $i = 1, \ldots, n$, and that we are given UMSs $\mathcal{A}_i$ and $\mathcal{A}_i'$ for each i.*

If for each $i = 1, \ldots, n$, $\mathbf{Exec}\,\mathcal{A}_i'$ is a $(\mathbf{i}_i, \mathbf{o}_i)$-refinement of $\mathbf{Exec}\,\mathcal{A}_i$ where $\mathbf{i}_i = \mathsf{Att}_{\mathcal{C}_i}^{\mathcal{A}}$ and $\mathbf{o}_i = \mathbf{i}_i \cup \{\mathsf{finished}_{\mathbf{Exec}\,\mathcal{A}_i}\}$ then $\mathcal{A}(\mathbf{Exec}\,\mathcal{A}_1', \ldots, \mathbf{Exec}\,\mathcal{A}_n')$ is a refinement of $\mathcal{A}(\mathbf{Exec}\,\mathcal{A}_1, \ldots, \mathbf{Exec}\,\mathcal{A}_n)$.

Proof. Suppose we are given a parameterized UMS $\mathcal{A}(\mathcal{Y}_1, \ldots, \mathcal{Y}_n)$, where the activity variable $\mathcal{Y}_i$ belongs to the component $\mathcal{C}_i$, for each $i = 1, \ldots, n$, and that we are given UMSs $\mathcal{A}_i$ and $\mathcal{A}_i'$ for each i. Suppose that $\mathbf{Exec}\,\mathcal{A}_i'$ is a $(\mathbf{i}_i, \mathbf{o}_i)$-refinement of $\mathbf{Exec}\,\mathcal{A}_i$ where $\mathbf{i}_i = \mathsf{Att}_{\mathcal{C}_i}^{\mathcal{A}}$ and $\mathbf{o}_i = \mathbf{i}_i \cup \{\mathsf{finished}_{\mathbf{Exec}\,\mathcal{A}_i}\}$, for each $i = 1, \ldots, n$. We have to show that $\mathcal{A}(\mathbf{Exec}\,\mathcal{A}_1', \ldots, \mathbf{Exec}\,\mathcal{A}_n')$ is a refinement of $\mathcal{A}(\mathbf{Exec}\,\mathcal{A}_1, \ldots, \mathbf{Exec}\,\mathcal{A}_n)$.

Firstly, we have $\mathsf{Msgs}_{\mathcal{A}(\mathcal{A}_1, \ldots, \mathcal{A}_n)} = \mathsf{Msgs}_{\mathcal{A}} = \mathsf{Msgs}_{\mathcal{A}(\mathcal{A}_1', \ldots, \mathcal{A}_n')}$ by construction.

Secondly, we have the bijections mapping $\mathcal{A}_i$ to $\mathcal{A}_i'$ for each i and being the identity on the other activities; they fulfill the required conditions by supposition on the $\mathcal{A}_i$ and $\mathcal{A}_i'$.

Theorem 7.13. *Refinement of UMSs is a precongruence with respect to composition by system formation.*

Proof. This follows from Facts 7.11 and 7.12.

Corollary 7.15. *Equivalence of UMSs is a congruence with respect to composition by system formation.*

Proof. This follows from Theorem 7.13.

Theorem 7.17. *Each UMS $\mathcal{A}$ is a $\mathcal{I}d$-interface refinement of itself, where $\mathcal{I}d(\mathcal{Y}) \stackrel{\text{def}}{=} \mathcal{Y}$.*

For all UMSs $\mathcal{A}$, $\mathcal{A}'$, and $\mathcal{A}''$ such that $\mathcal{A}'$ is a $\mathcal{I}$-interface refinement of $\mathcal{A}$ and $\mathcal{A}''$ is a $\mathcal{I}'$-interface refinement of $\mathcal{A}'$, $\mathcal{A}''$ is a $\mathcal{I}' \circ \mathcal{I}$-interface refinement of $\mathcal{A}$, where $\mathcal{I}' \circ \mathcal{I}(\mathcal{Y}) \stackrel{\text{def}}{=} \mathcal{I}'(\mathcal{I}(\mathcal{Y}))$.

Proof. Suppose we have a UMS $\mathcal{A}$ and define $\mathcal{I}d(\mathcal{Y}) \stackrel{\text{def}}{=} \mathcal{Y}$. Then we have $\mathcal{I}d(\mathcal{A}) = \mathcal{A}$ which is a refinement of $\mathcal{A}$ by reflexivity of refinement (see Theorem 7.13). Thus $\mathcal{A}$ is a $\mathcal{I}d$-interface refinement of itself.

Suppose we have UMSs $\mathcal{A}$, $\mathcal{A}'$, and $\mathcal{A}''$ such that $\mathcal{A}'$ is a $\mathcal{I}$-interface refinement of $\mathcal{A}$ and $\mathcal{A}''$ is a $\mathcal{I}'$-interface refinement of $\mathcal{A}'$, and define $\mathcal{I}' \circ \mathcal{I}(\mathcal{Y}) \stackrel{\text{def}}{=} \mathcal{I}'(\mathcal{I}(\mathcal{Y}))$. Then we have $\mathcal{I}' \circ \mathcal{I}(\mathcal{A}) = \mathcal{I}'(\mathcal{I}(\mathcal{A}))$. By assumption, we know that $\mathcal{A}'$ is a refinement of $\mathcal{I}(\mathcal{A})$ and that $\mathcal{A}''$ is a refinement of $\mathcal{I}'(\mathcal{A}')$. By substitutivity of refinement, we derive that $\mathcal{I}'(\mathcal{A}')$ is a refinement of $\mathcal{I}'(\mathcal{I}(\mathcal{A}))$, and by transitivity of refinement, this implies that $\mathcal{A}''$ is a refinement of $\mathcal{I}'(\mathcal{I}(\mathcal{A}))$ (see Theorem 7.13). Thus $\mathcal{A}''$ is a $\mathcal{I}' \circ \mathcal{I}$-interface refinement of $\mathcal{A}$.

C.3 Rely-Guarantee Specifications

Theorem 7.19. *Suppose that the UML Machine A fulfills the rely-guarantee specification (R, G) where $R{\frown}\mathcal{E} = R$ and $G{\frown}\mathcal{E} = G$, and suppose $E = \{\mathbf{I} : \mathbf{I}{\frown}\mathcal{E} = \mathbf{I}\}$.*

If the UML Machine A' $\mathcal{E}$-refines A and A' fulfills the rely-guarantee specification (R, E) then A' fulfills the rely-guarantee specification (R, G).

If the UML Machine A' delayed $\mathcal{E}$-refines A, G is stutter-closed, and A' fulfills the rely-guarantee specification (R, E), then A' fulfills the rely-guarantee specification (R, G).

Proof. Suppose that the UML Machine A fulfills the rely-guarantee specification (R, G) and the UML Machine A' $\mathcal{E}$-refines A, with $R{\frown}\mathcal{E} = R$ and $G{\frown}\mathcal{E} = G$. We need to show that A' fulfills the rely-guarantee specification (R, G). Suppose we are given $\mathbf{I} \in R$. We need to show that $[\![A']\!](\mathbf{I}) \subseteq G$. By assumption on A, we know that $[\![A]\!](\mathbf{I}) \subseteq G$. By assumption on A', we have $[\![A']\!](\mathbf{I}) \subseteq [\![A']\!](\mathbf{I}){\frown}\mathcal{E} \subseteq [\![A]\!](\mathbf{I}){\frown}\mathcal{E}$. Thus we may conclude that $[\![A']\!](\mathbf{I}) \subseteq G$, as required, since $G{\frown}\mathcal{E} = G$.

The proof for delayed refinement is analogous, using the fact that G is stutter-closed to conclude that $[\![A']\!](\mathbf{I}) \subseteq G$ from the fact that $[\![A']\!](\mathbf{I}) \subseteq\!\!\!\sim [\![A]\!](\mathbf{I}) \subseteq G$.

C.4 Reasoning About Security Properties

Fact 7.21. *Suppose we are given a UMS $\mathcal{A}$, an adversary $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$ of type A, and an execution $\mathbf{e} \in \mathbf{Run}\,\mathcal{A}_{adv}$. Then after execution of* $\mathbf{e}$, knows *evaluates to* $\mathcal{K}^{\mathbf{e}}_{adv}(\mathcal{A})$.

Proof. Suppose the execution $\mathbf{e}$ of $\mathcal{A}_{adv}$ has been executed. We need to show that an expression E is in the set which is the value of knows after execution of $\mathbf{e}$ if and only if $E \in \mathcal{K}^{\mathbf{e}}_{adv}(\mathcal{A})$. This is, however, the case by definition of $\mathcal{K}^{\mathbf{e}}_{adv}(\mathcal{A})$.

Fact 7.22. *Given a UMS $\mathcal{A}$ and an adversary $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$ of type A, the set* knows *of $\mathcal{A}_{adv}$ evaluates to a subset of $\mathcal{K}_A(\mathcal{A})$, at any point.*

Proof. Suppose we are given a system $\mathcal{A}$ and an expression $E \in$ knows. We need to show that E evaluates to an element of $\mathcal{K}_A(\mathcal{A})$. Suppose we are given an adversary $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$ and an execution $\mathbf{e}$ of $\mathcal{A}_{adv}$. By Fact 7.21, we know that after execution of $\mathbf{e}$, E evaluates to an element of $\mathcal{K}^{\mathbf{e}}_{adv}(\mathcal{A})$. Therefore, E evaluates to an element of $\mathcal{K}_A(\mathcal{A})$, by definition of $\mathcal{K}_A(\mathcal{A})$.

Fact 7.23. *Given a UMS $\mathcal{A}$ with a name v and an adversary $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$ of type A, then during any run $\mathbf{e} \in \mathbf{Run}\,\mathcal{A}_{adv}$, the name v evaluates to an element of $\mathcal{I}_A(\mathcal{A}, v)$, at any point.*

Proof. Suppose we are given a UMS $\mathcal{A}$ with a name v, an adversary $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$ of type A, and an expression $E \in \mathbf{Exp}$ which is a value of the name v after an execution $\mathbf{e}$ of $\mathcal{A}$. We need to show that $E \in \mathcal{I}_A(\mathcal{A}, v)$. By the definition of $\mathcal{I}^{\mathbf{e}}_{adv}(\mathcal{A}, v)$, we know that $E \in \mathcal{I}^{\mathbf{e}}_{adv}(\mathcal{A}, v)$. Therefore, $E \in \mathcal{I}_A(\mathcal{A}, v)$, by definition of $\mathcal{I}_A(\mathcal{A}, v)$.

Fact 7.25. *Suppose we are given UMSs $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B}$ is a refinement of $\mathcal{A}$, and an adversary type A, such that the accessible knowledge for A in $\mathcal{B}$ is no larger than that in $\mathcal{A}$. Then the UMS $\mathcal{B}$ is a black-box refinement in presence of adversaries of type A of the UMS $\mathcal{A}$.*

Proof. Suppose we are given UMSs $\mathcal{B}$ and $\mathcal{A}$ such that $\mathcal{B}$ is a refinement of $\mathcal{A}$, and an adversary adv' of a given type A. We need to show that there exists an adversary adv the UML Machine $\mathcal{B}_{adv'}$ is a black-box refinement of the UML Machine $\mathcal{A}_{adv}$.

Since the link structures of $\mathcal{A}$ and $\mathcal{B}$ are the same by definition of refinement for UMSs, it is sufficient to show that each possible sequence of contents of the family $(\mathsf{linkQu}_{\mathcal{B}}(l))_{l \in \mathsf{Links}_{\mathcal{B}}}$ of multi-set names is also a possible sequence of contents of the family $(\mathsf{linkQu}_{\mathcal{A}}(l))_{l \in \mathsf{Links}_{\mathcal{A}}}$. This follows from the assumption that $\mathcal{B}$ is a refinement of $\mathcal{A}$ and from the definition of refinement of UMSs, which implies that there are bijections b and $b_{\mathcal{C}}$ as in Definition 7.9 such that for each component $\mathcal{C} \in \mathsf{Comp}_{\mathcal{A}}$ and activity $A \in \mathsf{Act}^{\mathcal{A}}_{\mathcal{C}}$, the UML Machine of $b_{\mathcal{C}}(A)$ is a $(\mathbf{i}, \mathbf{o})$-refinement of the A Machine where $\mathbf{i} = \mathsf{Att}^{\mathcal{A}}_{\mathcal{C}}$ and $\mathbf{o} = \mathbf{i} \cup \{\mathsf{finished}_A\}$.

Theorem 7.27. *A UMS $\mathcal{A}$ preserves the secrecy of E against adversaries of type A if and only if $E \notin \mathcal{K}_A(\mathcal{A})$.*

Proof. Suppose that we are given a UMS $\mathcal{A}$, an expression $E \in \mathbf{Exp}$, and an adversary type A.

Firstly, we show that if $\mathcal{A}$ preserves the secrecy of E against adversaries of type A then $E \notin \mathcal{K}_A(\mathcal{A})$. We proceed by contraposition. We assume that we have $E \in \mathcal{K}_A(\mathcal{A})$. We need to show that $\mathcal{A}$ does not preserve the secrecy of E against adversaries of type A. By definition of preservation of secrecy, it is sufficient to show that there is an adversary $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$, an input sequence $\mathbf{i}$, and a sequence $s \in [\![\mathcal{A}_{adv}]\!]_{\emptyset,\{\mathsf{knows}\}}(\mathbf{i})$ such that one of the knowledge sets in s contains E. By the assumption $E \in \mathcal{K}_A(\mathcal{A})$ and the definition of $\mathcal{K}_A(\mathcal{A})$, we know that we have $E \in \mathcal{K}_A^n(\mathcal{A})$ for some $n \in \mathbb{N}$. Thus there is an adversary adv and an execution $\mathbf{e}$ of length n such that $E \in \mathcal{K}^{\mathbf{e}}_{adv}(\mathcal{A})$. Thus $E \in \mathsf{knows}$ after the nth iteration of $\mathcal{A}_{adv}$.

Secondly, we show that if $E \notin \mathcal{K}_A(\mathcal{A})$ then $\mathcal{A}$ preserves the secrecy of E against adversaries of type A. Suppose that $E \notin \mathcal{K}_A(\mathcal{A})$. We need to show that $\mathcal{A}$ preserves the secrecy of E against adversaries of type A; that is, for every adversary adv of type A, input sequence $\mathbf{i}$, and sequence $s \in [\![\mathcal{A}_{adv}]\!]_{\emptyset,\{\mathsf{knows}\}}(\mathbf{I})$, the knowledge sets in s do not contain E. Suppose we are given such an adversary adv, input sequence $\mathbf{i}$, and a sequence $s \in [\![\mathcal{A}_{adv}]\!]_{\emptyset,\{\mathsf{knows}\}}(\mathbf{I})$. It follows from Fact 7.22 that the knowledge sets in s do not contain E, since we have $E \notin \mathcal{K}_A(\mathcal{A})$ by assumption on E.

Theorem 7.28. *If the UMS $\mathcal{A}$ preserves the secrecy of E from adversaries of type A and the UMS $\mathcal{B}$ (delayed) refines $\mathcal{A}$, such that the accessible knowledge for A in $\mathcal{B}$ is no larger than that in $\mathcal{A}$, then $\mathcal{B}$ preserves the secrecy of E from adversaries of type A given inputs in $\mathcal{E}$.*

Proof. Suppose we are given a UMS $\mathcal{A}$ that preserves the secrecy of a given expression E from adversaries of type A given inputs in $\mathcal{E}$ for $\mathcal{E} \subseteq \mathbf{Events}$. Suppose that the UMS $\mathcal{B}$ refines $\mathcal{A}$. We need to show that $\mathcal{B}$ preserves the secrecy of E from adversaries of type A given inputs in $\mathcal{E}$.

Suppose we are given an adversary $adv \in \mathsf{Advers}_{\mathcal{B}}(A)$ and an execution $\mathbf{e}$ of $\mathcal{B}_{adv}$. Since $\mathcal{B}$ refines $\mathcal{A}$, we have $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$, and $\mathbf{e}$ is an execution of $\mathcal{A}_{adv}$, as far as observable to the adversary (up to stutter-equivalence, in the delayed case). Since $\mathcal{A}$ is assumed to preserve the secrecy of E, we conclude that $\mathcal{B}$ does, as well.

Theorem 7.29. *If the UMS $\mathcal{A}$ preserves the secrecy of E from adversaries of type A and the UMS $\mathcal{B}$ is a black-box refinement in presence of adversaries of type A of the UMS $\mathcal{A}$ then $\mathcal{B}$ preserves the secrecy of E from adversaries of type A.*

Proof. This directly follows from the Definition 7.24 of *black-box refinement in presence of adversaries.*

Theorem 7.31. *Each UMS $\mathcal{A}$ preserves the integrity of a variable v with respect to a set $E \subseteq \mathbf{Exp}$ of acceptable expressions against adversaries of type A if $\mathcal{I}_A(\mathcal{A}, v) \subseteq E$.*

Proof. Suppose that we are given a UMS $\mathcal{A}$, a variable v, a set $E \subseteq \mathbf{Exp}$ of acceptable expressions, and an adversary type A. We show that if $\mathcal{I}_A(\mathcal{A}, v) \subseteq E$ then $\mathcal{A}$ preserves the integrity of v respect to E against adversaries of type A. Suppose that $\mathcal{I}_A(\mathcal{A}, v) \subseteq E$. To show that $\mathcal{A}$ preserves the integrity of v with respect to E, it is sufficient to show that for every adversary *adv* of type A and every input sequence $\mathbf{i}$, v does not contain a value $a \notin E$ at any point. Suppose we are given such an adversary *adv* and an input sequence $\mathbf{i}$ such that at some point, v has the value a. By Fact 7.23 we can conclude that $a \in E$, by assumption on $\mathcal{I}_A(\mathcal{A}, v)$.

Theorem 7.32. *Suppose we are given UMSs $\mathcal{A}$ and $\mathcal{B}$. Suppose that $\mathcal{A}$ preserves the integrity of v with respect to a set $E \subseteq \mathbf{Exp}$ of acceptable expressions from adversaries of type A given inputs in $\mathcal{E}$ and that the UMS $\mathcal{B}$ is a $(\emptyset, \{v\})$-black-box refinement in presence of adversaries of type A of the UMS $\mathcal{A}$. Then $\mathcal{B}$ preserves the integrity of v with respect to E from adversaries of type A given inputs in $\mathcal{E}$.*

Proof. Suppose we are given a UMS $\mathcal{A}$ that preserves the integrity of a given expression v with respect to a set $E \subseteq \mathbf{Exp}$ of acceptable expressions from adversaries of type A given inputs in $\mathcal{E}$ for $\mathcal{E} \subseteq \mathbf{Events}$. Suppose that the UMS $\mathcal{B}$ $(\emptyset, \{v\})$-refines the UMS $\mathcal{A}$. We need to show that $\mathcal{B}$ preserves the integrity of v with respect to a set $E \subseteq \mathbf{Exp}$ of acceptable expressions from adversaries of type A given inputs in $\mathcal{E}$.

Suppose we are given $adv \in \mathsf{Advers}_{\mathcal{B}}(A)$ and an input sequence $\mathbf{i}$. We need to show that at no point of the execution of $\mathcal{B}_{adv}$ on the inputs $\mathbf{i}$, v takes on a value not contained in E.

Since $\mathcal{B}$ $(\emptyset, \{v\})$-refines the UMS $\mathcal{A}$. we have $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$, and any value of v in $\mathcal{B}$ is also a value of v in $\mathcal{A}$. Since $\mathcal{A}$ is assumed to preserve the integrity of v, we conclude that at no point of the execution of $\mathcal{B}_{adv}$ on the inputs $\mathbf{i}$, v takes on a value not contained in E.

Theorem 7.34. *Suppose we are given UMSs $\mathcal{A}$ and $\mathcal{B}$. Suppose that $\mathcal{A}$ provides authenticity of v with respect to its origin o from adversaries of type A given inputs in $\mathcal{E}$ and that the UMS $\mathcal{B}$ is a $(\emptyset, \{v, o\})$-white-box refinement of the UMS $\mathcal{A}$, such that the accessible knowledge for A in $\mathcal{B}$ is no larger than that in $\mathcal{A}$. Then $\mathcal{B}$ provides authenticity of v with respect to its origin o from adversaries of type A given inputs in $\mathcal{E}$.*

Proof. Suppose we are given UMSs $\mathcal{A}$ and $\mathcal{B}$. Suppose that $\mathcal{A}$ provides authenticity of v with respect to its origin o from adversaries of type A given inputs in $\mathcal{E}$ and that the UMS $\mathcal{B}$ is a $(\emptyset, \{v, o\})$-refinement of the UMS $\mathcal{A}$.

We would like to show that $\mathcal{B}$ provides authenticity of v with respect to its origin o from adversaries of type A given inputs in $\mathcal{E}$, which means that for

all $adv \in \mathsf{Advers}_{\mathcal{B}}(A)$ and each input sequence $\mathbf{i}$ whose multi-sets only contain elements in $\mathcal{E}$, at any point of the execution of $\mathcal{B}_{adv}$ on the inputs $\mathbf{i}$, v takes on a value which appeared first within the execution outQu_o, of all output queues and link queues in $\mathcal{B}$.

Suppose we are given an adversary $adv \in \mathsf{Advers}_{\mathcal{B}}(A)$ and an input sequence $\mathbf{i}$ whose multi-sets only contain elements in $\mathcal{E}$. Since $\mathcal{B}$ is a $(\emptyset, \{v, o\})$-refinement of $\mathcal{A}$, we have $adv \in \mathsf{Advers}_{\mathcal{A}}(A)$. Since $\mathcal{A}$ provides authenticity of v with respect to its origin o from adversaries of type A given inputs in $\mathcal{E}$, we know that at any point of the execution of $\mathcal{A}_{adv}$ on the inputs $\mathbf{i}$, v takes on a value which appeared first within the execution outQu_o, of all output queues and link queues in $\mathcal{A}$. Again, since $\mathcal{B}$ is a $(\emptyset, \{v, o\})$-refinement of $\mathcal{A}$, this means that that at any point of the execution of $\mathcal{B}_{adv}$ on the inputs $\mathbf{i}$, v takes on a value which appeared first within the execution outQu_o, of all output queues and link queues in $\mathcal{B}$.

Fact 7.38. *For any expression $E \in \mathbf{Data} \cup \mathbf{Var} \cup \mathbf{Keys}$ and any set of expressions $\mathcal{E}$, E is independent of $\mathcal{E}$ if there exists no expression $E' \in \mathcal{E}$ such that E is a subexpression of E'.*

Proof. Suppose we are given E and $\mathcal{E}$ as above such that there exists no expression E' of which E is a subexpression. We show that E is independent of $\mathcal{E}$, that is E is not an element of the subalgebra $\mathcal{A}$ of **Exp** generated by $\mathcal{E}$. $\mathcal{A}$ is defined to be the subset of values **Exp** obtained by recursively applying all operations of **Exp** starting with the set $\mathcal{E}$.

For each set of expressions $A \subseteq \mathbf{Exp}$ let $p(A)$ be the property that there exists no expression E' in A such that E is a subexpression of E'. We prove inductively that $E \notin \mathcal{A}$ by showing that $p(\mathcal{E})$ holds and that the validity of $p(A)$ is preserved by applying the operations of **Exp** pointwise to A:

- We have $p(\mathcal{E})$ by assumption.
- Assuming $p(A)$, we show by contraposition that for all $a_1, a_2 \in A$, E is not a subexpression of $a_1 :: a_2$. Suppose E is a subexpression of $a_1 :: a_2$ for some $a_1, a_2 \in A$. Without loss of generality, suppose that E is not a subexpression of a_1. Thus there exists a term t_1 which is equal to a_1 in **Exp** such that E is not a subterm of t_1. However, by assumption, E is a subterm of $t_1 :: t_2$ for every term t_2 which is equal to a_2 in **Exp**. Since $E \in \mathbf{Data} \cup \mathbf{Var} \cup \mathbf{Keys}$ by assumption, E is thus a subterm of every such t_2, by definition of the equations in **Exp**. Thus, E is a subexpression of a_2, by definition of subexpression.
- Suppose $p(A)$ holds. Then for every $a \in A$, E is not a subexpression of $\mathbf{head}(a)$. If E was a subterm of every term h that is equal to $\mathbf{head}(a)$ in A, then E is also a subterm of every term t that is equal to a in A, because the head of every such term t is such an h. An analogous argument applies to $\mathbf{tail}(\_)$.
- The cases for $\{\_\}\_$, $\mathcal{S}ign\_(\_)$, and $\mathcal{H}ash(\_)$ can be treated analogously to the one for $\_ :: \_$. For $\mathcal{D}ec\_(\_)$ and $\mathcal{E}xt\_(\_)$ one needs to choose a_1, a_2, t_1, t_2 such that t_1 and t_2 are minimal in length.

Fact 7.39. *For any expression* $E \in \mathbf{Data} \cup \mathbf{Var} \cup \mathbf{Keys}$ *and any set of expressions* $\mathcal{E} \subseteq \mathbf{Data} \cup \mathbf{Var} \cup \mathbf{Keys}$, E *is independent of* $\mathcal{E}$ *if and only if* $E \notin \mathcal{E}$.

Proof. This follows from Fact 7.38 since for $E, E' \in \mathbf{Data} \cup \mathbf{Var} \cup \mathbf{Keys}$, E is a subexpression of E' only if $E = E'$.

Fact 7.40. *Suppose we are given an atomic value* $d \in \mathbf{Data} \cup \mathbf{Keys}$ *in a UMS* $\mathcal{A}$ *which is fresh within a component* $\mathcal{D}$ *contained in* $\mathcal{A}$, *an adversary type* A *which does not have access to* $\mathcal{D}$ *and does not have* d *in its set* $\mathcal{K}_A^p$ *of previous knowledge, and an adversary* adv *of type* A. *Then during any run* $\mathbf{e} \in \mathbf{Run}\,\mathcal{A}_{adv}$, *if at any state* S *in* $\mathbf{e}$ *an output or link queue outside* $\mathcal{D}$ *contains* d *as a subexpressions, then there exists a state* S' *preceding* S *in* $\mathbf{e}$ *where* $\mathsf{outQu}_{\mathcal{D}}$ *contains* d *as a subexpressions.*

Proof. Suppose we are given an atomic value $d \in \mathbf{Data} \cup \mathbf{Keys}$ in a UMS $\mathcal{A}$ which is fresh within a component $\mathcal{D}$ contained in $\mathcal{A}$, an adversary type A which does not have access to $\mathcal{D}$, and an adversary adv of type A. We would like to show that during any run $\mathbf{e} \in \mathbf{Run}\,\mathcal{A}_{adv}$, if at any state S in $\mathbf{e}$ an output or link queue outside $\mathcal{D}$ contains d as a subexpressions, then there exists a state S' preceding S in $\mathbf{e}$ where $\mathsf{outQu}_{\mathcal{D}}$ contains d as a subexpressions.

Assume that we a given a run $\mathbf{e} \in \mathbf{Run}\,\mathcal{A}_{adv}$ and the first state S in $\mathbf{e}$ such that $\mathsf{outQu}_{\mathcal{D}}$ or an output or link queue outside $\mathcal{D}$ contains d as a subexpression. It is sufficient to show that in S, d is contained in $\mathsf{outQu}_{\mathcal{D}}$ and not an output or link queue outside $\mathcal{D}$, as a subexpression. According to the behavior of UMSs defined in Sect. 7.2, d can only appear as a subexpression in $\mathsf{outQu}_{\mathcal{D}}$ or in an output or link queue outside $\mathcal{D}$ after being inserted in a link queue by the adversary or after being output by a part of $\mathcal{A}$. By assumption on the capabilities and previous knowledge of the adverary, and since S was assumed to be the first state of its kind, the first possibility is ruled out. By the freshness assumption on d, and again by the assumption on S, the possibility left is that in S, d is contained in $\mathsf{outQu}_{\mathcal{D}}$ and not an output or link queue outside $\mathcal{D}$, as a subexpression.

Theorem 7.42. *Suppose that the UML Machine* A *prevents down-flow (resp. up-flow) with respect to the set* $H \subseteq \mathbf{MsgNm}$ *and that the UML Machine* B *refines* A. *Then* B *prevents down-flow (resp. up-flow) with respect to* H.

Proof. Suppose we are given UML Machines A, B and a set of message names $H \subseteq \mathbf{MsgNm}$, such that A prevents down-flow with respect to H and that B refines A.

We have to show that B prevents down-flow with respect to H. Suppose that we are given input sequences $\mathbf{i}, \mathbf{j}$ of event multi-sets and output sequences $\mathbf{o} \in [\![B]\!](\mathbf{i})$ and $\mathbf{p} \in [\![B]\!](\mathbf{j})$ with $\mathbf{i}_H = \mathbf{j}_H$. We have to show that $\mathbf{o}_H = \mathbf{p}_H$. Since B refines A, we know that $\mathbf{o} \in [\![A]\!](\mathbf{i})$ and $\mathbf{p} \in [\![A]\!](\mathbf{j})$. Since A prevents down-flow with respect to H, this implies that $\mathbf{o}_H = \mathbf{p}_H$.

The case for prevention of down-flow is analogous.

C.5 Formal Systems Development with UML

Fact 8.1. *During each given execution of a UML specification, each occurrence of a message is created at at most one location in the specification.*

Proof. We observe that the only ways in which a message can be newly introduced into the communication queues of the UMS is via a rule $\mathbf{ActionRuleSC}_S(a)$ for a *call* or *send* action a or a rule $\mathbf{ActionRuleSD}(msg)$ for a message msg: there is no usage in the formal semantics of the macro $\mathsf{tooutQu}()$ except in these rules, the macro $\mathsf{toinQu}()$ is not used at all, and there are messages added directly to the multi-sets inQu, outQu, or $\mathsf{linkQu}()$. Also, in the definition of the behavior of UMSs in terms of UML Machines in Sect. 7.2, no messages are newly added to the communication queues (but only transferred between the queues).

Thus to any occurrence m_i of a message m in any of the input, output, or link queues of the UMS modeling the specification at a given point of its execution, we can associate an occurrence l_i of a *call* or *send* action in a statechart or a message sent out in a sequence diagram, such that the occurrence m_i of m originated from the occurrence l_i of the action (for $i = 1, 2$). Conversely, each such occurrence l_i, when executed, adds a new occurrence m_i of m to the communication queues, by the definition of the associated action rule.

To see that each occurrence of a message is created at at most one location, it is sufficient to see that no occurrence of a message is removed during the execution of the UMS except when messages are consumed by its recipient.

For this we observe that in the formal semantics defined in this chapter, an occurrence of a message is only removed from the communication queues of a UML specification when it is consumed while it fires a transition at its recipient. Also, in the definition of the behavior of UMSs in terms of UML Machines in Sect. 7.2, no messages are removed from the communication queues (but only transferred between the queues).

Thus each occurence of a message is created at at most one location.

Fact 8.3. *(Delayed) $\mathcal{E}$-$(\mathbf{i}, \mathbf{o})$-black-box refinement of UML subsystems is a preorder for each set of events $\mathcal{E} \subseteq$ **Events** and tuples $\mathbf{i}$ and $\mathbf{o}$ of input and output names.*

Proof. This follows from Fact 7.8.

Fact 8.5. *The UML subsystem $\mathcal{S}'$ is a white-box refinement of the UML subsystem $\mathcal{S}$ then $\mathcal{S}'$ is also a black-box refinement of $\mathcal{S}$.*

Proof. This follows from Fact 7.10.

Theorem 8.6. *White-box refinement of UML subsystems is a precongruence with respect to composition by subsystem formation.*

Proof. This follows from Theorem 7.13.

Corollary 8.8. *White-box equivalence of UML subsystems is a congruence with respect to composition by subsystem formation.*

Proof. This follows from Theorem 8.6.

Theorem 8.10. *Each UML subsystem $\mathcal{S}$ is a $\mathcal{I}d$-interface refinement of itself, where $\mathcal{I}d(\mathcal{Y}) \stackrel{\text{def}}{=} \mathcal{Y}$.*

For all UML subsystems $\mathcal{S}$, $\mathcal{S}'$, and $\mathcal{S}''$ such that $\mathcal{S}'$ is a $\mathcal{I}$-interface refinement of $\mathcal{S}$ and $\mathcal{S}''$ is a $\mathcal{I}'$-interface refinement of $\mathcal{S}'$, $\mathcal{S}''$ is a $\mathcal{I}' \circ \mathcal{I}$-interface refinement of $\mathcal{S}$, where $\mathcal{I}' \circ \mathcal{I}(\mathcal{Y}) \stackrel{\text{def}}{=} \mathcal{I}'(\mathcal{I}(\mathcal{Y}))$.

Proof. This follows from Theorem 7.17.

Theorem 8.12. *Suppose that the UML subsystem $\mathcal{S}$ fulfills the rely-guarantee specification (R, G) and that $R \curvearrowright \mathcal{E} = R$ and $S \curvearrowright \mathcal{E} = S$.*

If the UML subsystem $\mathcal{S}'$ $\mathcal{E}$-black-box refines $\mathcal{S}$ then $\mathcal{S}'$ fulfills the rely-guarantee specification (R, G).

If the UML subsystem $\mathcal{S}'$ delayed $\mathcal{E}$-black-box refines A and G is stutter-closed then $\mathcal{S}'$ fulfills the rely-guarantee specification (R, G).

Proof. This follows from Theorem 7.19.

Fact 8.14. *Suppose we are given UML subsystems $\mathcal{S}$ and $\mathcal{T}$ such that $\mathcal{T}$ is a refinement of $\mathcal{S}$, and an adversary type A. Then the UMS $\mathcal{B}$ is a black-box refinement in presence of adversaries of type A of the UMS $\mathcal{A}$.*

Proof. This follows from Theorem 7.25.

Theorem 8.16. *Suppose we are given UML subsystems $\mathcal{S}$ and $\mathcal{T}$.*

If the UML subsystem $\mathcal{S}$ preserves the secrecy of E from adversaries of type A and $\mathcal{T}$ is a black-box refinement in presence of adversaries of type A, or $\mathcal{T}$ (delayed) refines $\mathcal{S}$ given adversaries of type A such that the accessible knowledge for A in $\mathcal{B}$ is no larger than that in $\mathcal{A}$, then $\mathcal{T}$ preserves the secrecy.

If $\mathcal{S}$ preserves the integrity of v with respect to a set $E \subseteq \mathbf{Exp}$ of acceptable expressions from adversaries of type A given inputs in $\mathcal{E}$ and $\mathcal{T}$ $(\emptyset, \{v\})$-black-box refines $\mathcal{S}$ then $\mathcal{T}$ preserves the integrity of v.

If $\mathcal{S}$ provides authenticity of v with respect to its origin o from adversaries of type A given inputs in $\mathcal{E}$ and $\mathcal{T}$ is a $(\emptyset, \{v, o\})$-black-box refinement of $\mathcal{S}$ such that the accessible knowledge for A in $\mathcal{B}$ is no larger than that in $\mathcal{A}$, then $\mathcal{T}$ $\mathcal{B}$ provides authenticity.

*If $\mathcal{S}$ prevents down-flow (resp. up-flow) with respect to the set $H \subseteq$ **MsgNm** and $\mathcal{T}$ white-box refines $\mathcal{S}$ then $\mathcal{T}$ prevents down-flow (resp. up-flow) with respect to H.*

Proof. The results in this theorem follow from the corresponding results on UMSs in Theorems 7.29, 7.28, 7.32, 7.34, and 7.42.

C.6 Secure Channels

Proposition 5.1. *The subsystem $\mathcal{C}$ preserves the secrecy of the variable* d *from adversaries of type* $A = \mathsf{default}$ *with specified previous knowledge* $\mathcal{K}_A^{\mathsf{p}}$, *given inputs from* $\mathbf{Data} \setminus \mathcal{K}_A^{\mathsf{p}}$.

Note that, intuitively, this proposition is obvious, because the adversary cannot read the channels. We give the proof to illustrate how to apply the formal framework.

Proof. We have to show that for every expression E which is a value of d at any point, $\mathcal{C}$ preserves the secrecy of E.

We use Theorem 7.27. Since an adversary of type $\mathsf{default}$ cannot access any of the components or links in $\mathcal{C}$, we have $\mathcal{K}_{\mathsf{A}}(\mathcal{C}) = \mathcal{K}_A^0$ (because there is no read access), and d takes values only in $\mathbf{Exp} \setminus \mathcal{K}_A^0$ (because there is no write access).

Thus for every expression E which is a value of d at any point, $\mathcal{C}$ preserves the secrecy of E, by definition of preservation of secrecy.

Proposition 5.2. *The subsystem $\mathcal{C}'$ is a delayed black-box refinement of $\mathcal{C}$ in presence of adversaries of type* $A = \mathsf{default}$ *with*

$$\mathcal{K}_A^{\mathsf{p}} \cap (\{\mathsf{K}_{\mathsf{S}}^{-1}, \mathsf{K}_{\mathsf{R}}^{-1}\} \cup \{\mathsf{k}_n, \{x :: n\}_{\mathsf{k}_n} : x \in \mathbf{Exp} \wedge n \in \mathbb{N}\}) = \emptyset$$

and for which $\mathcal{S}ign_{\mathsf{K}_{\mathsf{R}}^{-1}}(\mathsf{k}' :: m) \in \mathcal{K}_A^{\mathsf{p}}$ *implies* $\mathsf{k}' = \mathsf{k}_m$ *for all* $m \in \mathbb{N}$ *and* $\mathsf{k}' \in \mathbf{Exp}$.

Proof. We have to show that for every adversary b of type A for the UMS $[\![\mathcal{C}']\!]$ there exists an adversary a of type A for the UMS $[\![\mathcal{C}]\!]$ such that the derived UML Machine $\mathbf{Exec}\,[\![\mathcal{C}']\!]_b$ is a delayed black-box refinement of the UML Machine $\mathbf{Exec}\,[\![\mathcal{C}]\!]_a$.

Note that $\mathcal{K}_A(\mathcal{C}')$ is contained in the algebra generated by $\mathcal{K}_A^0 \cup \{\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{R}}^{-1}}(\mathsf{k_i} :: \mathsf{j})\}_{\mathsf{K_S}}\}$ and the expressions $\{\mathsf{d} :: n\}_{\mathsf{K}}$ for inputs d: Firstly, the adversary can obtain no certificate $\{\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{R}}^{-1}}(k :: \mathsf{j})\}_{\mathsf{K_S}}\}$ for $k \neq \mathsf{k_j}$, because the $\mathsf{Receiver}$ object only outputs the certificates $\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{R}}^{-1}}(\mathsf{k_j} :: \mathsf{j})\}_{\mathsf{K_S}}$ (for $\mathsf{j} \in \mathbb{N}$) to the Internet. Secondly, the sender outputs only messages of the form $\{\mathsf{d} :: n\}_k$ to the Internet, for inputs d and any $k \in \mathbf{Keys}$ for which a certificate $\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{R}}^{-1}}(k :: n)\}_{\mathsf{K_S}}$ has been received. Here k must be K_n since no other certificate can be produced (since the key $\mathsf{K}_{\mathsf{R}}^{-1}$ is never transmitted). Note also that $\mathcal{K}_{\mathcal{A}}^{\mathsf{p}} = \mathcal{K}_{\mathcal{A}}^0$ since there are no components accessed by the adversary.

Also, the values that an adversary for $\mathcal{C}'$ may insert into the Internet link may only delay the behavior of the two objects regarding $\mathsf{outQu}_{\mathcal{C}'}$ since the adversary has no other certificate signed with $\mathsf{K}_{\mathsf{R}}^{-1}$ and does not have access to the key $\mathsf{K}_{\mathsf{R}}^{-1}$, and because of the transaction numbers used. Thus any other value inserted is ignored by the two objects.

For any adversary b for $\mathcal{C}'$ we can thus derive an adversary a for $\mathcal{C}$ by omitting insert and read commands such that the UML Machine $\mathbf{Exec}\,[\![\mathcal{C}']\!]_b$

is a delayed black-box refinement of the UML Machine $\mathbf{Exec}\,[\![\mathcal{C}]\!]_a$ (since the outputs to $\mathsf{outQu}_{\mathcal{C}}$ (resp. $\mathsf{outQu}_{\mathcal{C}'}$) are stutter-equivalent).

Proposition 5.3. *The subsystem $\mathcal{C}'$ preserves the secrecy of the variable* d *from adversaries of type $A = \mathsf{default}$ with*

$$\mathcal{K}^{\mathsf{p}}_A \cap (\{\mathsf{K}_{\mathsf{S}}^{-1}, \mathsf{K}_{\mathsf{R}}^{-1}\} \cup \{\mathsf{k}_n, \{x :: n\}_{\mathsf{k}_n} : x \in \mathbf{Exp} \wedge n \in \mathbb{N}\}) = \emptyset$$

and for which $\mathcal{S}ign_{\mathsf{K}_{\mathsf{R}}^{-1}}(\mathsf{k}' :: m) \in \mathcal{K}^{\mathsf{p}}_A$ implies $\mathsf{k}' = \mathsf{k}_m$ for all $m \in \mathbb{N}$ and $\mathsf{k}' \in \mathbf{Exp}$.

Proof. Since $\mathcal{C}$ preserves the secrecy of the variable d from $\mathsf{default}$ adversaries given inputs from $\mathbf{Data} \setminus \mathcal{K}^{\mathsf{p}}_A$ by Proposition 5.1 and $\mathcal{C}'$ is a delayed black-box refinement of $\mathcal{C}$ given $\mathsf{default}$ adversaries with $\mathcal{K}^{\mathsf{p}}_A \cap (\{\mathsf{K}_{\mathsf{S}}^{-1}, \mathsf{K}_{\mathsf{R}}^{-1}, \mathsf{K}\} \cup \{\{x :: n\}_{\mathsf{K}} : x \in \mathbf{Exp} \wedge n \in \mathbb{N}\}) = \emptyset$ and for which $\mathcal{S}ign_{\mathsf{K}_{\mathsf{R}}^{-1}}(k') \in \mathcal{K}^{p}_A$ implies $\mathsf{K} = k'$ by Proposition 5.2, we can conclude that $\mathcal{C}'$ preserves the secrecy of the variable d from $\mathsf{default}$ adversaries with $\mathcal{K}^{\mathsf{p}}_A \cap (\{\mathsf{K}_{\mathsf{CA}}^{-1}, \mathsf{K}^{-1}\} \cup \{\{x :: n\}_{\mathsf{K}} : x \in \mathbf{Exp} \wedge n \in \mathbb{N}\}) = \emptyset$ and for which $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{R} :: \mathsf{k}') \in \mathcal{K}^{\mathsf{p}}_A$ implies $\mathsf{K} = k'$, given inputs from $\mathbf{Data} \setminus \mathcal{K}^{\mathsf{p}}_{\mathsf{A}}$, by Theorem 8.16.

C.7 A Variant of the Internet Protocol TLS

The Flaw

Theorem 5.4. *For given* C *and* i*, the UML subsystem $\mathcal{T}$ given in Fig. 5.3 does not preserve the secrecy of* $\mathsf{s_i}$ *from adversaries of type $A = \mathsf{default}$ with* $\{\mathsf{K_S}, \mathsf{K}_A, \mathsf{K}_A^{-1}\} \subseteq \mathcal{K}^{\mathsf{p}}_A$.

Proof. We show the existence of a successful attacker *adv*. We fix instances C and S with execution rounds i and j (where $\mathsf{S_i} = \mathsf{S}$) and denote the link between C and S as l_{CS}.

The adversary *adv* proceeds as follows:

- A message of the form $\mathsf{S.init}(\mathsf{N_i}, \mathsf{K_C}, \mathcal{S}ign_{\mathsf{K_C}^{-1}}(\mathsf{C} :: \mathsf{K_C}))$ in l_{CS} is replaced by the message $\mathsf{S.init}(\mathsf{N_i}, \mathsf{K}_A, \mathcal{S}ign_{\mathsf{K}_A^{-1}}(\mathsf{C} :: \mathsf{K}_A))$; that is, the public key $\mathsf{K_C}$ of C is replaced by the public key K_A of A at each occurrence and as the signature key.
- When S then sends back the message $\mathsf{resp}(\{\mathcal{S}ign_{\mathsf{K_S}^{-1}}(\mathsf{k_j} :: \mathsf{init}_1)\}_{\mathsf{K}_A}, \mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{S} :: \mathsf{K_S}))$, using K_A to encrypt the session key $\mathsf{k_j}$, *adv* can obtain $\mathsf{k_j}$ and replace the message by $\mathsf{resp}(\{\mathcal{S}ign_{\mathsf{K_S}^{-1}}(\mathsf{k_j} :: \mathsf{init}_1)\}_{\mathsf{K_C}}, \mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{S} :: \mathsf{K_S}))$.
- When C subsequently returns $\{\mathsf{s_i}\}_{\mathsf{k_j}}$, *adv* can extract the secret $\mathsf{s_i}$ (and forward the message).

An adversary machine that achieves this is defined as follows:

Rule Exec *adv* :
do – in – parallel
if $\mathsf{linkQu}_{\mathcal{T}}(l_{\mathsf{CS}}) = \{\!\!\{ e \}\!\!\} \wedge \mathbf{msgnm}(e) = \mathsf{S.init}$
then $\mathsf{linkQu}_{\mathcal{T}}(l_{\mathsf{CS}}) := \{\!\!\{ \mathsf{S.init}(\mathbf{Arg}_1(e), \mathsf{K}_A,$
$\mathcal{S}ign_{\mathsf{K}_A^{-1}}(\mathbf{fst}(\mathcal{E}xt_{\mathbf{Arg}_2(e)}(\mathbf{Arg}_3(\mathsf{e}))) :: \mathsf{K}_A)) \}\!\!\}$
if $\mathsf{linkQu}_{\mathcal{T}}(l_{\mathsf{CS}}) = \{\!\!\{ e \}\!\!\} \wedge \mathbf{msgnm}(e) = \mathsf{C.resp}$ **then**
do – in – parallel
$\mathsf{linkQu}_{\mathcal{T}}(l_{CS}) := \{\!\!\{ \mathsf{C.resp}(\{\mathcal{S}ign_{\mathsf{K}_A^{-1}}(\mathbf{Arg}_1(e))\}_{K_C}, \mathbf{Arg}_2(e)) \}\!\!\}$
$local := \{\!\!\{ \mathbf{fst}(\mathcal{E}xt_{K_S}(\mathcal{D}ec_{\mathsf{K}_A^{-1}}(\mathbf{Arg}_1(e)))) \}\!\!\}$
enddo
if $\mathsf{linkQu}_{\mathcal{T}}(l_{\mathsf{CS}}) = \{\!\!\{ e \}\!\!\} \wedge \mathbf{msgnm}(e) = \mathsf{S}.xchd$ **then**
$secret := \{\!\!\{ \mathcal{D}ec_{local}(\mathbf{Arg}_1(e)) \}\!\!\}$
enddo

Thus the adversary gets to know the secrets $\mathsf{C.s_i}$.

The Fix

Theorem 5.5. *Suppose we are given a particular execution of the repaired* TLS variant *subsystem $\mathcal{T}'$ (including all clients and servers), a client* C*, and a number I with $\mathsf{S} = \mathsf{S}_I$, and suppose that the server* S *is in its Jth execution round in the current execution when* C *in its Ith execution round initiates the protocol (that is, $\mathsf{C.i} = I$ and $\mathsf{S.j} = J$). Then this execution of $\mathcal{T}'$ preserves the secrecy of $\mathsf{C.s}_I$ against adversaries of type $A =$ default whose previous knowledge $\mathcal{K}_A^{\mathsf{p}}$ fulfills the following conditions:*

- *we have*

$$\Big(\{\mathsf{C.s}_I, \mathsf{K}_{\mathsf{C}}^{-1}, \mathsf{K}_{\mathsf{S}}^{-1}\} \cup \{\mathsf{S.k_j} : \mathsf{j} \geq J\} \cup \{\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{S}}^{-1}}(X :: \mathsf{C.N}_I :: \mathsf{K_C})\}_{\mathsf{K_C}} : X \in \mathbf{Keys}\}\Big) \cap \mathcal{K}_A^{\mathsf{p}} = \emptyset$$

- *for any $X \in \mathbf{Exp}$, $\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{C} :: X) \in \mathcal{K}_A^{\mathsf{p}}$ implies $X = \mathsf{K_C}$, and*
- *for any $X \in \mathbf{Exp}$, $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{S} :: X) \in \mathcal{K}_A^{\mathsf{p}}$ implies $X = \mathsf{K_S}$.*

Proof. We use Theorem 7.27 from Sect. 7.5.2. We show the following claim.

Claim. *For each knowledge set $\mathcal{K}_{adv}^{\mathsf{e}}(\mathcal{A})$ for an adversary adv of type A after an overall execution* **e** *of $\mathcal{T}'$, whose previous knowledge $\mathcal{K}_A^{\mathsf{p}}$ satisfies the conditions in the above statement of the theorem, there exists a subalgebra X_0 that is minimal with respect to the subset relation among the subalgebras X of* **Exp** *fulfilling the following two conditions,*[1] *such that X_0 contains $\mathcal{K}_{adv}^{\mathsf{e}}(\mathcal{A})$.*

Firstly, the following condition (1) is required to hold:

[1] $\mathcal{K}_{adv}^{\mathsf{e}}(\mathcal{A})$ is not contained in *every* such subalgebra X, because the actual messages exchanged may differ depending on the adversary behavior.

$$\mathcal{K}_A^{\mathsf{p}} \cup \Big\{ \mathsf{c}.\mathsf{N}_\mathsf{i}, \mathsf{K}_\mathsf{c}, \mathcal{S}ign_{\mathsf{K}_\mathsf{c}^{-1}}(\mathsf{c} :: \mathsf{K}_\mathsf{c}) : \mathsf{i} \in \mathbb{N} \wedge \mathsf{c} \in \mathsf{Client} \Big\}$$
$$\cup \Big\{ \mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{s} :: \mathsf{K}_\mathsf{s}) : \mathsf{s} \in \mathsf{Server} \wedge (\mathsf{s} = \mathsf{S} \Rightarrow \mathsf{K}_\mathsf{s} = \mathsf{K}_\mathsf{S}) \Big\}$$
$$\cup \Big\{ \{\mathsf{c}.\mathsf{s}_\mathsf{i}\}_\mathsf{k} : \mathsf{k} \in \mathbf{Keys} \wedge \mathsf{i} \in \mathbb{N} \wedge \mathsf{c} \in \mathsf{Client}$$
$$\wedge \exists \mathsf{K} \in \mathbf{Keys}, E \in \mathbf{Exp}, E' \in X$$
$$\Big(\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(E) \in X \wedge \mathbf{fst}(E) = \mathsf{c}.\mathsf{S}_\mathsf{i} \wedge \mathbf{snd}(E) = \mathsf{K}$$
$$\wedge \mathcal{E}xt_\mathsf{K}(\mathcal{D}ec_{\mathsf{K}_\mathsf{c}^{-1}}(E')) = (\mathsf{k}, \mathsf{c}.\mathsf{N}_\mathsf{i}, \mathsf{K}_\mathsf{c}) \Big) \Big\}$$
$$\subseteq X.$$

Condition (2) requires that for each $\mathsf{j} \in \mathbb{N}$ *and* $\mathsf{s} : \mathit{Server}$ *and for an associated fixed key* $\mathsf{k}_{\mathsf{j},\mathsf{s}} \in \mathbf{Keys} \cap X$*, a fixed expression* $x_{\mathsf{j},\mathsf{s}} \in \mathbf{Exp}$*, and a fixed nonce* $n_{\mathsf{j},\mathsf{s}} \in \mathbf{Data} \cap X$ *with* $\mathcal{S}ign_{\mathsf{k}_{\mathsf{j},\mathsf{s}}^{-1}}(x_{\mathsf{j},\mathsf{s}} :: \mathsf{k}_{\mathsf{j},\mathsf{s}}) \in X$,[2] *we have*

$$\{\mathcal{S}ign_{\mathsf{K}_\mathsf{s}^{-1}}(\mathsf{s}.\mathsf{k}_\mathsf{j} :: n_{\mathsf{j},\mathsf{s}} :: \mathsf{k}_{\mathsf{j},\mathsf{s}})\}_{\mathsf{k}_{\mathsf{j},\mathsf{s}}} \in X.$$

Note that in the second condition, it can be the case that $\mathsf{k}_{\mathsf{j},\mathsf{s}}^{-1} \in \mathcal{K}_{adv}^{\mathbf{e}}(\mathcal{A})$, but then $\mathsf{k}_{\mathsf{j},\mathsf{s}} \neq \mathsf{K}_\mathsf{c}$ for any client c (because $\mathsf{K}_\mathsf{C}^{-1} \notin \mathcal{K}_{adv}^{\mathbf{e}}(\mathcal{A})$ since $\mathsf{K}_\mathsf{c}^{-1} \notin \mathcal{K}_\mathsf{A}^{\mathsf{p}}$ and $\mathsf{K}_\mathsf{c}^{-1}$ is never sent out), and c will notice that something is wrong in the corrected protocol (and because the counter j is increased, the adversary cannot make the server publish another signature with the same k_j and the correct K_C).

Proof of claim. Intuitively, the above claim holds because each knowledge set $\mathcal{K}_{adv}^{\mathbf{e}}(\mathcal{A})$ is by definition the subalgebra of the algebra of expressions **Exp** built up from $\mathcal{K}_A^{\mathsf{p}}$ in interaction with the protocol participants during the protocol run $\mathbf{e}$. To argue in more detail, we have to consider what knowledge the adversary can gain from interaction with the protocol participants. From the first message of a client c, the adversary can learn the expressions $\mathsf{c}.\mathsf{N}_\mathsf{i}, \mathsf{K}_\mathsf{c}$, and $\mathcal{S}ign_{\mathsf{K}_\mathsf{c}^{-1}}(\mathsf{c} :: \mathsf{K}_\mathsf{c})$. From the first message of a server s, the adversary can firstly learn $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{s} :: \mathsf{K}_\mathsf{s})$. Secondly, for each encryption key $\mathsf{K} \in \mathbf{Keys}$ in the knowledge of the adversary such that the adversary knows $\mathcal{S}ign_{\mathsf{K}^{-1}}(x :: \mathsf{K})$ for some $x \in \mathbf{Exp}$, and for each N known to the adversary, the adversary learns $\{\mathcal{S}ign_{\mathsf{K}_\mathsf{s}^{-1}}(\mathsf{s}.\mathsf{k}_\mathsf{j} :: \mathsf{vN} :: \mathsf{K})\}_\mathsf{K} \in X$, but only a unique such expression for a given server s, protocol run $\mathbf{e}$, and transaction number j, because the transaction number j is increased as long as the protocol is iterated (this is reflected by the fact that X_0 is required to be minimal). From the second message from a client c, for each encryption key $\mathsf{K} \in \mathbf{Keys}$ such that

- $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(E)$ is known to the adversary for an $E \in \mathbf{Exp}$ with $\mathbf{fst}(E) = \mathsf{c}.\mathsf{S}_\mathsf{i}$ and $\mathbf{snd}(E) = \mathsf{K}$, and such that

[2] Note that condition (1) guarantees the existence of these unique expressions associated with each $\mathsf{j} \in \mathbb{N}$ and $\mathsf{s} : \mathit{Server}$.

- there exists $E' \in \mathbf{Exp}$ which is known to the adversary and such that $\mathcal{E}xt_{\mathsf{K}}(\mathcal{D}ec_{\mathsf{K_c}^{-1}}(E')) = (\mathsf{k}, \mathsf{c.N_i}, \mathsf{K_c})$ for some $\mathsf{k} \in \mathbf{Keys}$,

the adversary learns $\{\mathsf{c.s_i}\}_{\mathsf{K}} \in X$.

Since there are no other messages sent out by the specified system, the claim holds by the definition of the adversary knowledge as the algebra generated by the exchanged messages and the initial adversary knowledge. This completes proof of the claim.

Thus it is sufficient to show that $\mathsf{C.s}_I \notin X_0$ for every X_0 defined above, because $\mathcal{K}_A(\mathcal{A}) \stackrel{\text{def}}{=} \bigcup_{adv,\mathbf{e}} \mathcal{K}^{\mathbf{e}}_{adv}(\mathcal{A})$ is contained in the union of all such X_0 by the above argument. We aim for a contradiction by fixing such an X_0 and assuming that $\mathsf{C.s}_I \in X_0$. X_0 is defined to be a minimal subalgebra satisfying the conditions (1) and (2) above. Recall that from the definition of the algebra of expressions $\mathbf{Exp}$ in Sect. 3.3.3 as a free algebra it follows that $\mathsf{C.s}_I$ is different from any other expression not containing it, since no equation with such an expression is defined. In particular, we have $\mathsf{C.s}_I \neq \mathsf{c.s_i}$ for any client c and number i with $c \neq C$ or $\mathsf{i} \neq I$. Thus the only occurrence in the conditions defining X_0 in a minimal way, where $\mathsf{C.s}_I$ may be introduced as a subterm, is in the requirement that X_0 contains $\{\mathsf{C.s}_I\}_{\mathsf{k}}$ for each key $\mathsf{k} \in \mathbf{Keys}$ for which there exist $\mathsf{K} \in \mathbf{Keys}, E \in \mathbf{Exp}, E' \in X_0$ such that

$$\mathcal{S}ign_{\mathsf{K_{CA}^{-1}}}(E) \in X_0 \wedge \mathbf{fst}(E) = \mathsf{S} \wedge \mathbf{snd}(E) = \mathsf{K} \\ \wedge \mathcal{E}xt_{\mathsf{K}}(\mathcal{D}ec_{\mathsf{K_C^{-1}}}(E')) = (\mathsf{k}, \mathsf{C.N}_I, \mathsf{K_C})$$

in condition (1). The assumption $\mathsf{C.s}_I \in X_0$ thus implies that there exists a key $\mathsf{k} \in \mathbf{Keys}$ for which there exist $\mathsf{K} \in \mathbf{Keys}$, $E \in \mathbf{Exp}$, $E' \in X_0$ such that

$$\mathcal{S}ign_{\mathsf{K_{CA}^{-1}}}(E) \in X_0 \wedge \mathbf{fst}(E) = \mathsf{S} \wedge \mathbf{snd}(E) = \mathsf{K} \\ \wedge \mathcal{E}xt_{\mathsf{K}}(\mathcal{D}ec_{\mathsf{K_C^{-1}}}(E')) = (\mathsf{k}, \mathsf{C.N}_I, \mathsf{K_C}).$$

By definition of X_0 and assumption on $\mathcal{K}_A^{\mathsf{p}}$, the condition $\mathcal{S}ign_{\mathsf{K_{CA}^{-1}}}(E) \in X_0 \wedge \mathbf{fst}(E) = \mathsf{S} \wedge \mathbf{snd}(E) = \mathsf{K}$ implies that $\mathsf{K} = \mathsf{K_S}$ (since any expression of this form in $\mathcal{K}_A^{\mathsf{p}}$ must satisfy this, and also any such expression introduced in X_0). Similarly, $E' \in X_0$ with $\mathcal{E}xt_{\mathsf{K_S}}(\mathcal{D}ec_{\mathsf{K_C^{-1}}}(E')) = (\mathsf{k}, \mathsf{C.N_i}, \mathsf{K_C})$ implies $\mathsf{k} = \mathsf{S.k_j}$ for some j, because $E' \notin \mathcal{K}_A^{\mathsf{p}}$ by assumption on the previous adversary knowledge $\mathcal{K}_A^{\mathsf{p}}$, because $\mathsf{K_S^{-1}}$ is never communicated, and because the expression $\{\mathcal{S}ign_{\mathsf{K_S^{-1}}}(\mathsf{S.k_j} :: \mathsf{n_{j,s}} :: \mathsf{k_{j,s}})\}_{\mathsf{k_{j,s}}}$ (in condition (2)) is the only expression with a subterm of the form $\mathcal{S}ign_{\mathsf{K_S^{-1}}}(\mathsf{k} :: n_{\mathsf{j,s}} :: \mathsf{k_{j,s}})$ that is introduced (and we can also conclude that $n_{\mathsf{j,s}} = \mathsf{C.N}_I$ and $\mathsf{k_{j,s}} = \mathsf{K_C}$ in this term). Furthermore, we can conclude $\mathsf{j} \geq J$ by the assumption that S is in its Jth execution round when C is in its Ith round, and by the requirement that the $\mathsf{C.N_i}$ should be fresh (that is, each distinct from any other occurring value). Thus by assumption on the previous adversary knowledge $\mathcal{K}_A^{\mathsf{p}}$, we have $\mathsf{S.k_j} \notin \mathcal{K}_A^{\mathsf{p}}$ since $\mathsf{j} \geq J$, and thus the adversary must have learned $\mathsf{S.k_j}$ in a protocol interaction. By

the freshness assumption on $\mathsf{S.k_j}$, the only message containing $\mathsf{S.k_j}$ is a term of the form $\{\mathcal{S}ign_{\mathsf{K_S}^{-1}}(\mathsf{S.k_j} :: n_{\mathsf{j,s}} :: \mathsf{k_{j,s}})\}_{\mathsf{k_{j,s}}}$. By condition (2) and the minimality of X_0, we know that $n_{\mathsf{j,s}} = \mathsf{C.N_l}$ and $\mathsf{k_{j,s}} = \mathsf{K_C}$ for any such term by the above observation. Therefore, this term has to be decrypted with $\mathsf{K_C^{-1}}$ in order to get the $\mathsf{S.k_j}$. The only protocol participant that possesses $\mathsf{K_C^{-1}}$ and that could thus provide this service for the adversary is C (since the other participants do not have $\mathsf{K_C^{-1}}$ in their initial knowledge, and $\mathsf{K_C^{-1}}$ is never exchanged). However, none of the values in $\{\mathcal{S}ign_{\mathsf{K_S}^{-1}}(\mathsf{S.k_j} :: \mathsf{C.N}_I :: \mathsf{K_C})\}_{\mathsf{K_C}}$ is ever sent out to the network by C. Thus we must conclude that $\mathsf{K_C^{-1}} \in \mathcal{K}_A^{\mathsf{p}}$, which contradicts the initial assumption about $\mathcal{K}_A^{\mathsf{p}}$.

One can see as follows that the adversary knowledge before each *iteration* of the system satisfies these conditions as well:

(1) In the Ith execution round of the client C, no data of the form $X.s_i$ is output except $\mathsf{C.s}_I$, which, as the theorem shows, is kept secret from the adversary. The secret keys $\mathsf{K_C^{-1}}, \mathsf{K_S^{-1}}$ (for each C, S) are never output at all. The key $\mathsf{S.K}_J$ is only sent out during the Jth executing round of S, and it follows from the above theorem that in that round, the key is not leaked to the adversary (because otherwise the adversary would gain knowledge of $\mathsf{C.s}_I$ by decrypting the contents of the xchd message). Similarly, an expression of the form $\{\mathcal{S}ign_{\mathsf{K_S}^{-1}}(X :: \mathsf{C.N}_I :: \mathsf{K_C})\}_{\mathsf{K_C}}$ (for $X \in \mathbf{Keys}$) is only output in the Ith execution round of C (and is of no use in any later round).
(2) For any $X \in \mathbf{Exp}$, $\mathcal{S}ign_{\mathsf{K_C}^{-1}}(\mathsf{C} :: X)$ is sent out only for $X = \mathsf{K_C}$ (and $\mathsf{K_C^{-1}}$ is not sent out at all).
(3) For any $X \in \mathbf{Exp}$, $\mathcal{S}ign_{\mathsf{K_{CA}}^{-1}}(\mathsf{S} :: X)$ is sent out only for $X = \mathsf{K_S}$ (and $\mathsf{K_{CA}^{-1}}$ is not sent out at all).

Corollary 5.6. *Any execution of $\mathcal{T}'$ over all clients and servers and all execution rounds preserves the secrecy of each $\mathsf{C.s}_I$ (for $\mathsf{C} : \mathsf{Client}$ and $1 \leq I \leq \mathsf{l}$) against adversaries of type $A = \mathsf{default}$ whose previous knowledge $\mathcal{K}_A^{\mathsf{p}}$ before the overall execution of $\mathcal{T}'$ fulfills the following conditions:*

- *we have*

$$\Big(\{\mathsf{K_c^{-1}}, \mathsf{K_s^{-1}}, \mathsf{c.s_i}, \mathsf{s.k_j}, \{\mathcal{S}ign_{\mathsf{K_s}^{-1}}(X :: \mathsf{c.N_i} :: \mathsf{K_c})\}_{\mathsf{K_c}} : \mathsf{c} : \mathsf{Client} \wedge \mathsf{s} : \mathsf{Server} \wedge 1 \leq \mathsf{i} \leq \mathsf{l} \wedge 1 \leq \mathsf{j} \wedge X \in \mathbf{Keys}\}\Big) \cap \mathcal{K}_A^{\mathsf{p}} = \emptyset,$$

- *for any $X \in \mathbf{Exp}$ and any $\mathsf{c} : \mathsf{Client}$, $\mathcal{S}ign_{\mathsf{K_c}^{-1}}(\mathsf{c} :: X) \in \mathcal{K}_A^{\mathsf{p}}$ implies $X = \mathsf{K_c}$, and*
- *for any $X \in \mathbf{Exp}$ and any $\mathsf{s} : \mathsf{Server}$, $\mathcal{S}ign_{\mathsf{K_{CA}}^{-1}}(\mathsf{s} :: X) \in \mathcal{K}_A^{\mathsf{p}}$ implies $X = \mathsf{K_s}$.*

Proof. Suppose we are given an execution $\mathbf{e}$ of $\mathcal{T}'$, a client C, and a number I. Then we have $\mathsf{S}_I = \mathsf{S}$ for a server S, and within the execution $\mathbf{e}$, at the

point where $\mathsf{C.i} = I$, we have $\mathsf{S.j} = J$ for a number J. Since the conditions on the previous adversary knowledge in the current corollary imply those of the previous theorem, we can thus directly apply the theorem.

C.8 Common Electronic Purse Specifications

C.8.1 Purchase Transaction

Vulnerability

Theorem 5.7. *$\mathcal{P}$ does not provide* merchant security *against* insider *adversaries with* $\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{ID}_{\mathsf{C}'} :: \mathsf{K}_{\mathsf{C}'}), \mathsf{K}_{\mathsf{C}'}^{-1}\} \subseteq \mathcal{K}_A^{\mathsf{p}}$.

Proof. We show the existence of a successful attacker *adv*. We assume that the adversary has a certificate $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{ID}_{\mathsf{C}'} :: \mathsf{K}_{\mathsf{C}'})$ and the corresponding private key $\mathsf{K}_{\mathsf{C}'}^{-1}$ (this should of course not be linked to the identity of the adversary to avoid identification). We write l_{CP} (resp. l_{PD}) for the link between C and P (resp. P and D). $l_{\mathsf{AP}'}$ is a link between the attacker and the PSAM P′. Thus:

Rule Exec *adv* :
if $\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}}) = \{\!\{e\}\!\} \wedge \mathbf{msgnm}(e) = \mathsf{P.Ccert}$
then
$\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{AP}'}) := \mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}})$;
$\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}}) := \{\!\{\mathsf{P.Ccert}(\mathsf{ID}_{\mathsf{C}'}, \mathsf{K}_{\mathsf{C}'}, \mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{ID}_{\mathsf{C}'} :: \mathsf{K}_{\mathsf{C}'}))\}\!\}$
else
if $\mathsf{linkQu}_{\mathcal{P}}(l_{CP}) = \{\!\{e\}\!\} \wedge \mathbf{msgnm}(e) = \mathsf{C.Deb}$
then $m := \mathbf{fst}(\mathcal{D}ec_{\mathsf{K}_{\mathsf{C}'}^{-1}}(\mathbf{Arg}_2(e)))$;
if $\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{AP}'}) = \{\!\{e\}\!\} \wedge \mathbf{msgnm}(e) \in \{\mathsf{C.Pcert}, \mathsf{C.Deb}\}$
then $\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}}) := \mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{AP}'})$;
if $\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}}) = \{\!\{e\}\!\} \wedge \mathbf{msgnm}(e) = \mathsf{P}'.\mathsf{Resp}$
then do $-$ **in** $-$ **parallel**
$\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{AP}'}) := \mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}})$
$\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{DP}}) := \{\!\{\mathsf{D.Disp}(m)\}\!\}$
enddo

Note that again we give a simplified presentation of the UML Machine for increased readability. For example, according to the definition of an adversary in Sect. 7.5, the command $\mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{AP}'}) := \mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}})$ has to be realized by using commands of the form $\mathsf{read}_{l_{\mathsf{CP}}}(m) \equiv m := \mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{CP}})$ and $\mathsf{insert}_{l_{\mathsf{AP}'}}(e) \equiv \mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{AP}'}) := \mathsf{linkQu}_{\mathcal{P}}(l_{\mathsf{AP}'}) \uplus \{\!\{e\}\!\}$, in a suitable iteration.

We explain how the attacker UML Machine proceeds. If a message with name P.Ccert is sent over l_{CP}, the adversary copies it to $l_{AP'}$ and replaces it in l_{CP} by $\mathsf{P.Ccert}(\mathsf{ID}_{\mathsf{C}'}, \mathsf{K}_{\mathsf{C}'}, \mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{ID}_{\mathsf{C}'} :: \mathsf{K}_{\mathsf{C}'}))$. Otherwise, if a message with name C.Deb is sent over l_{CP}, the adversary extracts the amount $\mathbf{fst}(\mathcal{D}ec_{\mathsf{K}_{\mathsf{C}'}^{-1}}(\mathbf{Arg}_2(e)))$ from it and stores it in m. A message with name C.Pcert

or $\mathsf{C.Deb}$ in $l_{\mathsf{AP}'}$ is copied to l_{CP}. If l_{CP} consists of a message with name $\mathsf{P}'.\mathsf{Resp}$, the content of l_{CP} is copied to $l_{\mathsf{AP}'}$ and the message $\mathsf{D.Disp}(m)$ is sent to l_{DP}.

The above condition of *merchant security* is clearly violated: when executing $\mathcal{P}$ in the presence of *adv*, D receives the value M_{NT}, but P is not in possession of $\mathcal{S}ign_{\mathsf{K}_\mathsf{C}^{-1}}(\mathsf{ID_C}::\mathsf{ID_P}::\mathsf{M_{NT}}::\mathsf{NT})$.

Proposed Solution

Proposition 5.8. *$\mathcal{P}'$ provides secrecy of $\mathsf{K}_\mathsf{C}^{-1}$, $\mathsf{K}_\mathsf{P}^{-1}$ and integrity of $\mathsf{K}_\mathsf{C}^{-1}$, $\mathsf{K_C}$, $\mathsf{K_{CA}}$, $\mathsf{ID_C}$, $\mathsf{K}_\mathsf{P}^{-1}$, $\mathsf{K_P}$, $\mathsf{M_{NT}}$, $\mathsf{SK_{NT}}$, NT (meaning that the adversary should not be able to make the atttributes take on values previously known only to him) against* insider *adversaries with $\mathcal{K}_\mathsf{A}^\mathsf{p} \cap \{\mathsf{K}_\mathsf{C}^{-1}, \mathsf{K}_\mathsf{P}^{-1}\} = \emptyset$.*

Proof. For an adversary to gain knowledge of $\mathsf{K}_\mathsf{C}^{-1}$, $\mathsf{K}_\mathsf{P}^{-1}$, the adversary would have to read these expressions from one of the two communication links. We therefore have to consider, if at any point any of the two expressions is communicated over any of the two communication links. According to the specification, none of the values is output by any of the protocol participants at any time. Therefore secrecy of $\mathsf{K}_\mathsf{C}^{-1}$, $\mathsf{K}_\mathsf{P}^{-1}$ is provided since these values are never sent outside the smart cards (which under the current threat scenario are assumed to be impenetrable).

For the adversary to violate the integrity of any of the attributes $\mathsf{K}_\mathsf{C}^{-1}$, $\mathsf{K_C}$, $\mathsf{K_{CA}}$, $\mathsf{ID_C}$, $\mathsf{K}_\mathsf{P}^{-1}$, $\mathsf{K_P}$, $\mathsf{M_{NT}}$, $\mathsf{SK_{NT}}$, the adversary would have to cause their values to take on an atomic value in $\mathbf{Data}^a$, during the interaction with the protocol participants. In particular, their values would have to change. From the protocol specification, we can see that the value of none of these attributes is changed at all during the execution of the protocol. Thus their integrity is preserved.

Similarly, for the adversary to violate the integrity of the attribute NT, the adversary would have to cause its value to take on an atomic value in $\mathbf{Data}^a$, during the interaction with the protocol participants. From the protocol specification, we can see that the value of NT is changed only to take on values of the form 0, $0+1$, $0+1+1$, etc., all of which are not in $\mathbf{Data}^a$. Thus the integrity of NT is preserved.

Theorem 5.9. *Consider adversaries of type $A = \mathsf{insider}$ with*

$$\begin{aligned}\mathcal{K}_A^\mathsf{p} \cap \Big(& \{\mathsf{K}_\mathsf{C}^{-1}, \mathsf{K}_\mathsf{P}^{-1}, \mathsf{K}_\mathsf{CA}^{-1}\} \cup \{\mathsf{SK_{NT}} : \mathsf{NT} \in \mathbb{N}\} \\ & \cup \{\mathcal{S}ign_{\mathsf{K}_\mathsf{P}^{-1}}(E) : E \in \mathbf{Exp}\} \cup \{\mathcal{S}ign_{\mathsf{K}_\mathsf{C}^{-1}}(E) : E \in \mathbf{Exp}\} \\ & \cup \{\mathcal{S}ign_{\mathsf{SK_{NT}}}(E) : E \in \mathbf{Exp} \wedge \mathsf{NT} \in \mathbb{N}\}\Big) = \emptyset\end{aligned}$$

and such that for each $X \in \mathbf{Exp}$ with $\mathcal{S}ign_{\mathsf{K}_\mathsf{CA}^{-1}}(X::K) \in \mathcal{K}_\mathsf{A}^\mathsf{p}$, $X = \mathsf{ID_C}$ implies $K = \mathsf{K_C}$ and $X = \mathsf{ID_P}$ implies $K = \mathsf{K_P}$. The following security guarantees are provided by $\mathcal{P}'$ in the presence of adversaries of type A:

Cardholder security: For all $\mathsf{ID_C}, \mathsf{ID_P}, \mathsf{M_{NT}}, \mathsf{NT}, \mathsf{K_C^{-1}}$ *such that* $\mathsf{K_C}$ *is valid for* $\mathsf{ID_C}$*, if* P *is in possession of* $\mathcal{S}ign_{\mathsf{K_C^{-1}}}(\mathsf{ID_C} :: \mathsf{ID_P} :: \mathsf{M_{NT}} :: \mathsf{NT})$ *then* C *is in possession of* $\mathcal{S}ign_{\mathsf{K_P^{-1}}}(\mathsf{M_{NT}} :: \mathsf{SK_{NT}} :: \mathsf{ID_P} :: \mathsf{ID_C} :: \mathsf{NT})$ *(for some* $\mathsf{SK_{NT}}$ *and* $\mathsf{K_P^{-1}}$ *such that the corresponding key* $\mathsf{K_P}$ *is valid for* $\mathsf{ID_P}$*).*

Merchant security: Each time D *receives the value* $\mathsf{M_{NT}}$*,* P *is in possession of* $\mathcal{S}ign_{\mathsf{K_{CA}^{-1}}}(\mathsf{ID_C} :: \mathsf{K_C})$ *and* $\mathcal{S}ign_{\mathsf{K_C^{-1}}}(\mathsf{ID_C} :: \mathsf{ID_P} :: \mathsf{M_{NT}} :: \mathsf{NT})$ *for some* $\mathsf{ID_C}$*,* $\mathsf{K_C^{-1}}$*, and a new value* NT*.*

Card issuer security: After each completed purchase transaction, let S *be the sum of all* $\mathsf{M_{NT}}$ *in the sequence consisting of the processed elements of the form* $\mathcal{S}ign_{\mathsf{K_C^{-1}}}(\mathsf{ID_C} :: \mathsf{ID_P} :: \mathsf{M_{NT}} :: \mathsf{NT})$ *(with possibly varying* $\mathsf{ID_C}$*,* $\mathsf{ID_P}$*, and* $\mathsf{K_C^{-1}}$*, such that the corresponding key* $\mathsf{K_C}$ *is valid for* $\mathsf{ID_C}$ *and where the* NT *are mutually distinct for fixed* C*). Also, let* S' *be the sum of all* $\mathsf{M'_{NT'}}$ *in the sequence of processed* $\mathcal{S}ign_{\mathsf{K_{P'}^{-1}}}(\mathsf{M'_{NT'}} :: \mathsf{SK'_{NT'}} :: \mathsf{ID_{C'}} :: \mathsf{ID_{P'}} :: \mathsf{NT'})$ *(with possibly varying* $\mathsf{ID_{C'}}, \mathsf{ID_{P'}}$*, and* $\mathsf{K_{P'}^{-1}}$*, such that the corresponding key* $\mathsf{K_{P'}}$ *is valid for* $\mathsf{ID_{P'}}$*, and where the* $\mathsf{NT'}$ *are mutually distinct for fixed* $\mathsf{C'}$*). Then* S *is no greater than* S'*.*

Proof

Cardholder security: We proceed by contraposition. Suppose that (for any $\mathsf{SK_{NT}}, \mathsf{K_P^{-1}}$ such that the corresponding key $\mathsf{K_P}$ is valid for $\mathsf{ID_P}$) C is not in possession of $\mathcal{S}ign_{\mathsf{K_P^{-1}}}(\mathsf{M_{NT}} :: \mathsf{SK_{NT}} :: \mathsf{ID_P} :: \mathsf{ID_C} :: \mathsf{NT})$. We would like to show that for every $\mathsf{K_C^{-1}}$ such that the corresponding key $\mathsf{K_C}$ is valid for $\mathsf{ID_C}$, P is not in possession of $\mathcal{S}ign_{\mathsf{K_C^{-1}}}(\mathsf{ID_C} :: \mathsf{ID_P} :: \mathsf{M_{NT}} :: \mathsf{NT})$. We fix such $\mathsf{ID_C}$, $\mathsf{K_C}$, and $\mathsf{K_C^{-1}}$.

We consider:

- the joint knowledge set $\mathcal{K}$ of all participants except C (that is, the objects P and D, and any given adversary, which according to the threat scenario are not able to penetrate the smart card on which C resides) and
- the knowledge set $\mathcal{K}_C$ of C.

Claim. $\mathcal{K}$ *is contained in every subalgebra* X *of* **Exp** *containing*

$$
\begin{aligned}
&\mathbf{Keys} \setminus \{\mathsf{K_C^{-1}}\} \cup \mathcal{K}_A^{\mathsf{p}} \cup \mathbf{Data} \cup \\
&\Big\{ \{\mathcal{S}ign_{\mathsf{K_C^{-1}}}(\mathsf{ID_C} :: \mathsf{id_P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}}, \\
&\qquad \mathcal{S}ign_{\mathsf{sk}}(\mathsf{m} :: \{\mathcal{S}ign_{\mathsf{K_C^{-1}}}(\mathsf{ID_C} :: \mathsf{id_P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}}) : \\
&\qquad\qquad \mathsf{id_P}, \mathsf{k_P}, \mathsf{m}, \mathsf{sk}, \mathsf{nt}, \mathsf{E} \in \mathcal{K}_{\mathsf{C}} \wedge \mathcal{S}ign_{\mathsf{K_{CA}^{-1}}}(\mathsf{id_P} :: \mathsf{k_P}) \in \mathcal{K}_{\mathsf{C}} \\
&\qquad\qquad \wedge \mathcal{E}xt_{\mathsf{k_P}}(E) = \mathsf{m} :: \mathsf{sk} :: \mathsf{id_P} :: \mathsf{ID_C} :: \mathsf{nt} \Big\}.
\end{aligned}
$$

Note that $\mathcal{S}ign_{\mathsf{sk}}(\mathsf{m} :: \{\mathcal{S}ign_{\mathsf{K_C^{-1}}}(\mathsf{ID_C} :: \mathsf{id_P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}})$ is actually redundant, but included for explicitness. Note also that it is not claimed that $\mathcal{K}$ is actually

the intersection of such algebras. For example, any of the above algebras (and thus their intersection) contains the key $\mathsf{K}_{\mathsf{CA}}^{-1}$, although $\mathcal{K}$ does not. The latter fact is nevertheless used in the proof (below when using the claim). A similar remark applies to terms of the form $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{ID} :: \mathsf{K})$. Note that $\mathcal{K}$ contains $\mathsf{SK}_{\mathsf{NT}}$, but not $\mathsf{K}_{\mathsf{C}}^{-1}$ (as shown below).

The above claim holds because the knowledge set $\mathcal{K}$ is by definition the subalgebra of the algebra of expressions **Exp** built up from the initial knowledge by the protocol participants except C and any adversary in interaction with C. We thus have to consider what knowledge the other participants can gain from interaction with C. The expressions learned from the first message from C are contained in X because X is assumed to contain all keys $K \in \mathbf{Keys} \setminus \{\mathsf{K}_{\mathsf{C}}^{-1}\}$, and all data in **Data**. The expressions learned from the second message from C are contained in X because X is assumed to contain $\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{id}_\mathsf{P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}}$ and $\mathcal{S}ign_{\mathsf{sk}}(\mathsf{m} :: \{\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{id}_\mathsf{P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}})$ for all $\mathsf{id}_\mathsf{P}, \mathsf{k}_\mathsf{P} \in \mathcal{K}_\mathsf{C}$ with $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{id}_\mathsf{P} :: \mathsf{k}_\mathsf{P}) \in \mathcal{K}_\mathsf{C}$ and $m, \mathsf{sk}, \mathsf{nt}, E \in \mathcal{K}_C$ with $\mathcal{E}xt_{k_P}(E) = m :: \mathsf{sk} :: \mathsf{id}_P :: \mathsf{ID}_C :: \mathsf{nt}$, and because C must receive the values $\mathsf{id}_P, k_P, \mathcal{S}ign_{K_{\mathsf{CA}}^{-1}}(\mathsf{id}_P :: k_P), m, \mathsf{sk}, \mathsf{nt}, E$ before sending out the messages $\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{id}_\mathsf{P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}}$ and $\mathcal{S}ign_{\mathsf{sk}}(\mathsf{m} :: \{\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{id}_\mathsf{P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}})$.

In particular, we have $\mathsf{K}_{\mathsf{C}}^{-1} \notin \mathcal{K}$, because the initial knowledge of P, D, and the adversary does not include $\mathsf{K}_{\mathsf{C}}^{-1}$, and it (or anything it could be derived from) is not transmitted.

Under the above assumption that $\mathcal{S}ign_{\mathsf{K}_{\mathsf{P}}^{-1}}(\mathsf{M}_{\mathsf{NT}} :: \mathsf{SK}_{\mathsf{NT}} :: \mathsf{ID}_\mathsf{P} :: \mathsf{ID}_\mathsf{C} :: \mathsf{NT}) \notin \mathcal{K}_\mathsf{C}$ (for any $\mathsf{SK}_{\mathsf{NT}}, \mathsf{K}_{\mathsf{P}}^{-1}$ such that the corresponding key K_P is valid for ID_P), we prove that such a subalgebra X with $\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{ID}_\mathsf{P} :: \mathsf{M}_{\mathsf{NT}} :: \mathsf{NT}) \notin X$ exists. Let X be the **Exp** subalgebra generated by

$$
\begin{aligned}
G := \; & \mathbf{Keys} \setminus \{\mathsf{K}_{\mathsf{C}}^{-1}\} \cup \mathbf{Data} \cup \\
& \{\{\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{id}_\mathsf{C} :: \mathsf{id}_\mathsf{P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}}, \\
& \mathcal{S}ign_{\mathsf{sk}}(\mathsf{m} :: \{\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{id}_\mathsf{C} :: \mathsf{id}_\mathsf{P} :: \mathsf{m} :: \mathsf{nt})\}_{\mathsf{sk}}) : \\
& (\mathsf{id}_\mathsf{C}, \mathsf{id}_\mathsf{P}, \mathsf{m}, \mathsf{nt}) \neq (\mathsf{ID}_\mathsf{C}, \mathsf{ID}_\mathsf{P}, \mathsf{M}_{\mathsf{NT}}, \mathsf{NT})\}.
\end{aligned}
$$

By construction, X fulfills the above conditions, using the fact that the adversary does not have access to $\mathsf{K}_{\mathsf{CA}}^{-1}$ (since it is not in the adverary's initial knowledge and it (or anything it could be derived from) is never transmitted) and thus does not have access to terms of the form $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{id}_\mathsf{P} :: \mathsf{k}_\mathsf{P})$ unless k_P is valid for id_P. Also, we have $\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{ID}_\mathsf{P} :: \mathsf{M}_{\mathsf{NT}} :: \mathsf{NT}) \notin X$.

Thus we have $\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{ID}_\mathsf{P} :: \mathsf{M}_{\mathsf{NT}} :: \mathsf{NT}) \notin \mathcal{K}$.

Merchant security: Each time D receives the value M_{NT}, P is in possession of $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{K}_\mathsf{C})$ and $\mathcal{S}ign_{\mathsf{K}_{\mathsf{C}}^{-1}}(\mathsf{ID}_\mathsf{C} :: \mathsf{ID}_\mathsf{P} :: \mathsf{M}_{\mathsf{NT}} :: \mathsf{NT})$ for some ID_C, $\mathsf{K}_{\mathsf{C}}^{-1}$, and a new value NT.

By the specification of P (and the assumption of a secure communication link between P and D), D receives the value M_{NT} only after P has

checked the conditions in its part of the protocol; that is, P is in possession of $\mathcal{S}ign_{\mathsf{K}_{\mathsf{CA}}^{-1}}(\mathsf{id}_\mathsf{C} :: \mathsf{k}_\mathsf{C})$ and $\mathcal{S}ign_{\mathsf{K}_\mathsf{C}^{-1}}(\mathsf{id}_\mathsf{C} :: \mathsf{ID}_\mathsf{P} :: \mathsf{M}_{\mathsf{NT}} :: \mathsf{NT})$ for some id_C. Newness of NT in this expression is guaranteed since P creates the value itself by incrementing it between different runs of the protocol, and because the value is prevented from rolling over.

Card issuer security: This follows from the proof of *cardholder security.*

C.8.2 Load Transaction

Vulnerabilities

Theorem 5.10. $\mathcal{L}$ *does not provide* load acquirer security *against adversaries of type* insider *with* $\{\mathsf{cep}, \mathsf{lda}, \mathsf{m}_\mathsf{n}\} \subseteq \mathcal{K}_A^\mathsf{p}$.

Proof. An attacker may proceed as follows. The attack assumes a threat scenario where the attacker may be (or collaborate with) the card issuer. Thus it suffices to give a modification of the card issuer behavior that achieves the goal of the attack, that is to successfully complete the protocol and to possess a signature of the form ml_n but with the changed amount $\tilde{\mathsf{m}}$ in the end. The following modified card issuer specification J simply stores $\mathcal{S}ign_{\mathsf{r}'}(\mathsf{cep}'' :: \mathsf{nt}'' :: \mathsf{lda}'' :: \tilde{\mathsf{m}} :: \mathsf{s1}'' :: \mathsf{hc}'_{\mathsf{nt}} :: \mathsf{hl}' :: \mathsf{h2l}')$ instead of ml' in the logging object CLog:

Rule Exec J :
case $\mathsf{currState}_J$ **of**
Init: **do trans**$_J$(Load, (cep, lda, m, nt, s1, R, ml, hl, h2l),
valid(cep) $\wedge$ $\mathcal{E}xt_{\mathsf{K}_{\mathsf{CI}}}(\mathsf{s1}) = \mathsf{cep} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{nt}$
$\wedge$ $\mathcal{E}xt_{\mathcal{D}ec_{\mathsf{K}_{\mathsf{LI}}}(\mathsf{R})}(\mathsf{ml}) = \mathsf{cep} :: \mathsf{nt} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{s1} ::$
$\mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rc}_{\mathsf{nt}}) :: \mathsf{hl} :: \mathsf{h2l}$;
L.RespL($\mathcal{S}ign_{\mathsf{K}_{\mathsf{CI}}}(\mathsf{cep} :: \mathsf{n1} :: \mathsf{s1} :: \mathsf{hl})$), Load;
L.RespL(0), Fail)
Load: **do trans**$_J$(Comp, (cep, lda, m, nt, r2l, s3), *true*;
i.llog(cep, lda, m, nt, r, $\tilde{\mathsf{ml}}$, r2l), Final; ,)
Fail: **do trans**$_J$([], [], *true*; i.llog(cep, lda, 0, nt, r, $\tilde{\mathsf{ml}}$, 0), Final; ,)
Final: **do** finished := *true*

Here we use the macro

$$\tilde{\mathsf{ml}} \equiv \mathcal{S}ign_{\mathsf{r}}(\mathsf{cep} :: \mathsf{nt} :: \mathsf{lda} :: \tilde{\mathsf{m}} :: \mathsf{s1} :: \mathsf{hc}_{\mathsf{nt}} :: \mathsf{hl} :: \mathsf{h2l})$$

Proposed solution

Proposition 5.11. $\mathcal{L}'$ *provides secrecy of* $\mathsf{K}_{\mathsf{CI}}, \mathsf{K}_{\mathsf{L}}^{-1}, \mathsf{K}_{\mathsf{I}}^{-1}$ *and integrity of* K_{CI}, $\mathsf{K}_{\mathsf{L}}^{-1}$, $\mathsf{K}_{\mathsf{I}}^{-1}$, cep, nt, $\mathsf{rc}_{\mathsf{nt}}$, lda, n, rl_n, $\mathsf{r2l}_\mathsf{n}$, m_n *(meaning that the adversary should not be able to make the atttributes take on values previously known only to him) against* insider *adversaries with* $\mathcal{K}_A^\mathsf{p} \cap \{\mathsf{K}_{\mathsf{CI}}, \mathsf{K}_{\mathsf{L}}^{-1}, \mathsf{K}_{\mathsf{I}}^{-1}\} = \emptyset$.

Proof. Secrecy is evident since these values are never sent outside the smart cards (which are under the current threat scenario assumed to be impenetrable).

Similarly, integrity of $\mathsf{K_{CI}}$, $\mathsf{K_L^{-1}}$, $\mathsf{K_I^{-1}}$, cep, $\mathsf{rc_{nt}}$, lda, $\mathsf{rl_n}$, $\mathsf{r2l_n}$, $\mathsf{m_n}$ is evident since these values are not changed during the execution of the specification. Note that the secure definition of $\mathsf{m_{nt}}$ (which is outside the current specification) again relies on a secure connection between the terminal where the cash is entered and the LSAM. Also, the creation of the random values $\mathsf{rc_{nt}}$, $\mathsf{rl_n}$, $\mathsf{r2l_n}$ is outside the current scope. Finally, integrity of nt (resp. n) in the sense of Sect. 4.1.2 follows from the fact that the card (resp. the LSAM) changes the value of nt (resp. n) during the protocol irrespective of the behavior of the environment.

Theorem 5.12. *In the presence of adversaries of type* $A = \mathsf{insider}$ *with*

$$\mathcal{K}_A^{\mathsf{p}} \cap \{\mathsf{K_{CI}}, \mathsf{K_L^{-1}}, \mathsf{K_I^{-1}}\} \cup \{\mathsf{rc_{nt}} : \mathsf{nt} \in \mathbb{N}\} \cup \{\mathsf{rl_n}, \mathsf{r2l_n} : \mathsf{n} \in \mathbb{N}\} = \emptyset$$

the following security guarantees are provided by $\mathcal{L}'$*:*

Cardholder security: For any message $\mathsf{CLog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$ *sent to* $\mathsf{c} : \mathsf{CLog}$*, if* $\mathsf{m} \neq 0$ *(that is, the card seems to have been loaded with* m*) then* $\mathsf{rl} \neq 0$ *and*

$$\mathcal{E}xt_{\mathsf{K_{CI}}}(\mathsf{s2}) = \mathsf{cep} :: \mathsf{nt} :: \mathcal{S}ign_{\mathsf{K_{CI}}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{nt}) :: \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl})$$

holds (that is, the card issuer certifies rl *to be a valid proof for the transaction). For any two messages* $\mathsf{CLog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$ *and* $\mathsf{CLog}(\mathsf{lda}', \mathsf{m}', \mathsf{nt}', \mathsf{s2}', \mathsf{rl}')$ *sent to* $\mathsf{c} : \mathsf{CLog}$*, we have* $\mathsf{nt} \neq \mathsf{nt}'$*.*

Load acquirer security: Suppose that we have $\mathsf{ml_n} \in \mathcal{K}$ *and* $\mathsf{rl_n} \in \mathcal{K}$ *where* $\mathsf{ml_n} = \mathcal{S}ign_{\mathsf{K_L^{-1}}}(\mathsf{cep} :: \mathsf{nt} :: \mathsf{lda} :: \mathsf{m_n} :: \mathsf{s1} :: y :: \mathsf{hl_n} :: \mathsf{h2l_n})$ *with* $\mathsf{hl_n} = \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl_n})$ *and* $\mathsf{h2l_n} = \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{r2l_n})$*, for some* cep*,* nt*,* s1*, and* y*. Then at the end of an execution of* L *either of the following two conditions hold:*

- *a message* $\mathsf{Llog}(\mathsf{cep}, \mathsf{lda}, \mathsf{m_n}, \mathsf{nt}, x)$ *has been sent to* $\mathsf{I} : \mathsf{LLog}$ *(which implies that* L *has received and retains* $\mathsf{m_n}$ *in cash) or*
- *a message* $\mathsf{Llog}(\mathsf{cep}, \mathsf{lda}, 0, \mathsf{nt}, x)$ *has been sent to* $\mathsf{I} : \mathsf{LLog}$*, for some* x *(that is, the load acquirer assumes that the load failed and returns the amount* $\mathsf{m_n}$ *to the cardholder), and we have* $x' \in \mathcal{K}_\mathsf{L}$ *and* $z \in \mathcal{K}$ *with* $z = \mathcal{S}ign_{\mathsf{K_I^{-1}}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m_n} :: \mathsf{nt} :: y')$ *where* $y' = \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: x') = y$ *(that is, the load acquirer can prove that the load was aborted).*

Card issuer security: For each message $\mathsf{CLog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$ *sent to* $\mathsf{c} : \mathsf{CLog}$*, if* $\mathsf{m} \neq 0$ *and*

$$\mathcal{E}xt_{\mathsf{K_{CI}}}(\mathsf{s2}) = \mathsf{cep} :: \mathsf{nt} :: \mathcal{S}ign_{\mathsf{K_{CI}}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{nt}) :: \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl})$$

holds for some lda*, then the card issuer has a valid signature* $\mathsf{ml_n}$ *corresponding to this transaction.*

Proof.

Cardholder security: Suppose that the message $\mathsf{Clog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$ has been sent to $\mathsf{c} : \mathsf{CLog}$, where $\mathsf{m} \neq 0$. We need to show that $\mathsf{rl} \neq 0$ and that

$$\mathcal{E}xt_{\mathsf{K_{CI}}}(\mathsf{s2}) = \mathsf{cep} :: \mathsf{nt} :: \mathcal{S}ign_{\mathsf{K_{CI}}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{nt}) :: \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl})$$

holds. By assumption, the connection between $\mathsf{C} : \mathsf{Card}$ and $\mathsf{c} : \mathsf{CLog}$ is secure (since the objects are on the same smart card). This implies that C actually sent the message $\mathsf{Clog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$. According to the specification of C, this can only happen if $\mathsf{rl} \neq 0$ and if $\mathcal{E}xt_{\mathsf{K_{CI}}}(\mathsf{s2}) = \mathsf{cep} :: \mathsf{nt} :: \mathsf{s1} :: \mathsf{hl}$ holds, where $\mathsf{s1} = \mathcal{S}ign_{\mathsf{K_{CI}}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{nt})$ and $\mathsf{hl} = \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl})$.

Suppose the two messages $\mathsf{Clog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$ and $\mathsf{Clog}(\mathsf{lda}', \mathsf{m}', \mathsf{nt}', \mathsf{s2}', \mathsf{rl}')$ have been sent to $\mathsf{c} : \mathsf{CLog}$. We need to show that $\mathsf{nt} \neq \mathsf{nt}'$. Again, by the threat scenario we can conclude that C sent the two messages to c. Suppose without loss of generality that $\mathsf{Clog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$ was sent first. According to the statechart specification for C, C reaches the final state immediately afterwards. According to the overall activity diagram given in the specification, C starts a new protocol run only after nt is incremented (and rolling over is not possible). Thus we have $\mathsf{nt}' \geq \mathsf{nt} + 1$, in particular $\mathsf{nt} \neq \mathsf{nt}'$.

Load acquirer security: Suppose that we have $\mathsf{ml_n} \in \mathcal{K}$ and $\mathsf{rl_n} \in \mathcal{K}$ where $\mathsf{ml_n} = \mathcal{S}ign_{\mathsf{K_L}^{-1}}(\mathsf{cep} :: \mathsf{nt} :: \mathsf{lda} :: \mathsf{m_n} :: \mathsf{s1} :: y :: \mathsf{hl_n} :: \mathsf{h2l_n})$ with $\mathsf{hl_n} = \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl_n})$ and $\mathsf{h2l_n} = \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{r2l_n})$, for some cep, nt, $\mathsf{s1}$, and y, and that a message $\mathsf{Llog}(\mathsf{cep}, 0, \mathsf{nt}, x)$ has been sent to $\mathsf{l} : \mathsf{LLog}$, for some x. We need to show that there exist $x' \in \mathcal{K}_\mathsf{L}$ and $z \in \mathcal{K}$ with $z = \mathcal{S}ign_{\mathsf{K_l}^{-1}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m_n} :: \mathsf{nt} :: y')$ where $y' = \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: x') = y$.

By the assumed threat scenario, the communication link between L and l is secure (and according to the specification only L can send messages to l). This implies that the message $\mathsf{Llog}(\mathsf{cep}, 0, \mathsf{nt}, x)$ to $\mathsf{l} : \mathsf{LLog}$ originated at L. According to the specifications of L, this implies that L previously received a message $\mathsf{RespC}(\mathsf{s3}, x')$ with $x' = x$, $x' \neq 0$, and such that $\mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: x') = y'$ for a value y' received in the message $\mathsf{RespI}(\mathsf{cep}, \mathsf{nt}, \mathsf{s1}, y')$ previously in the same protocol run, and such that for the second argument of the message $\mathsf{RespL}(\mathsf{s2}, z)$ received immediately before $\mathsf{RespC}(\mathsf{s3}, x')$, $\mathcal{E}xt_{\mathsf{K_l}}(z) = \mathsf{cep} :: \mathsf{lda} :: \mathsf{m_n} :: \mathsf{nt} :: y'$ holds (in particular we have $x', z \in \mathcal{K}_\mathsf{L}$).

Card issuer security: Suppose that the message $\mathsf{Clog}(\mathsf{lda}, \mathsf{m}, \mathsf{nt}, \mathsf{s2}, \mathsf{rl})$ was sent to $\mathsf{c} : \mathsf{CLog}$, where $\mathsf{m} \neq 0$ and $\mathcal{E}xt_{\mathsf{K_{CI}}}(\mathsf{s2}) = \mathsf{cep} :: \mathsf{nt} :: \mathcal{S}ign_{\mathsf{K_{CI}}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{nt}) :: \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl})$ holds for some lda. We need to show that the card issuer has a valid signature $\mathsf{ml_n}$ corresponding to this transaction.

From the specification of C we see that C has received the message $\mathsf{Credit}(\mathsf{s2}, \mathsf{rl})$ just before in the same protocol run, and that $\mathcal{E}xt_{\mathsf{K_{CI}}}(\mathsf{s2}) = \mathsf{cep} :: \mathsf{nt} :: \mathsf{s1} :: \mathsf{hl}$ holds, where $\mathsf{s1} := \mathcal{S}ign_{\mathsf{K_{CI}}}(\mathsf{cep} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{nt})$ and $\mathsf{hl} := \mathcal{H}ash(\mathsf{lda} :: \mathsf{cep} :: \mathsf{nt} :: \mathsf{rl})$. Since the key $\mathsf{K_{CI}}$ is kept secret by C and I (see Proposition 5.11), we may conclude that I created $\mathsf{s2}$. According to the specification of I, this can only be the case if $\mathsf{ml} \in \mathcal{K}_\mathsf{I}$ with $\mathcal{E}xt_{\mathsf{K_L}}(\mathsf{ml}) = \mathsf{cep} :: \mathsf{nt} :: \mathsf{lda} :: \mathsf{m} :: \mathsf{s1} :: \widehat{\mathsf{hc}}_{\mathsf{nt}} :: \mathsf{hl} :: \mathsf{h2l}$.

References

[Aba00] M. Abadi. Security protocols and their properties. In F. L. Bauer and R. Steinbrüggen, editors, *Foundations of Secure Computation*, pages 39–60. IOS Press, Amsterdam, 2000. 20th International Summer School, Marktoberdorf, Germany.

[ABKL93] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and delegation with smart-cards. *Science of Computer Programming*, 21(2):93–113, 1993.

[Abr90] S. Abramsky. A generalized Kahn principle for abstract asynchronous networks. In *Mathematical foundations of programming semantics (New Orleans, LA, 1989)*, pages 1–21. Springer, Berlin Heidelberg New York, 1990.

[AdBD+02] J. Ø. Aagedal, F. den Braber, T. Dimitrakos, B. A. Gran, D. Raptis, and K. Stølen. Model-based risk assessment to improve enterprise security. In *6th International Enterprise Distributed Object Computing Conference (EDOC 2002)*, pages 51–62. IEEE Computer Society, New York, 2002.

[AFG02] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. *Information and Computation*, 174(1):37–83, 2002.

[AG99] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, January 1999.

[AGM00] S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors. *Handbook of logic in computer science*, volume 1–5, pages xii+827. The Clarendon Press, New York, 1992–2000.

[AGMO04] S. Abramsky, D. R. Ghica, A. S. Murawski, and C.-H. L. Ong. Applying game semantics to compositional software modeling and verification. In K. Jensen and A. Podelski, editors, *10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 421–435. Springer, Berlin Heidelberg New York, 2004.

[AJ01] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In N. Kobayashi and B. C. Pierce, editors, *Theoretical Aspects of Computer Software (4th International Symposium, TACS*

2001), volume 2215 of *Lecture Notes in Computer Science*, pages 82–94. Springer, Berlin Heidelberg New York, 2001.

[AJP95] M. Abrams, S. Jajodia, and H. Podell, editors. *Information security: an integrated collection of essays*. IEEE Computer Society Press, Silver Springs, MD, 1995. Download at `http://www.acsac.org/secshelf/book001/book001.html`.

[AJSW00] N. Asokan, P. Janson, M. Steiner, and M. Waidner. State of the art in electronic payment systems. *Advances in Computers*, 53:425–449, 2000.

[AK96] R. Anderson and M. Kuhn. Tamper resistance – a cautionary note. In *USENIX Workshop on Electronic Commerce*, pages 1–11, 1996.

[AKS96] T. Aslam, I. Krsul, and E. Spafford. Use of a taxonomy of security faults. In *19th National Information Systems Security Conference (NISSC)*, pages 551–560, 1996.

[AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, May 1991.

[AL93] M. Abadi and L. Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems 15*, 1:73–132, January 1993.

[Ald01] A. Aldini. Probabilistic information flow in a process algebra. In K. G. Larsen and M. Nielsen, editors, *12th International Conference on Concurrency Theory (CONCUR 2001)*, volume 2154 of *Lecture Notes in Computer Science*, pages 152–168. Springer, Berlin Heidelberg New York, 2001.

[AN96] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[And94] R. Anderson. Why cryptosystems fail. *Communications of the ACM*, 37(11):32–40, November 1994.

[And99] R. Anderson. The formal verification of a payment system. In M. Hinchey and J. Bowen, editors, *Industrial-Strength Formal Methods in Practice*, pages 43–52. Springer, Berlin Heidelberg New York, 1999.

[And01] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, New York, 2001.

[And02] R. Anderson. Two remarks on public key cryptology. Technical Report UCAM-CL-TR-549, University of Cambridge, 2002.

[APG95] M. Abrams, H. Podell, and D. Gambel. Security engineering. In Abrams et al [AJP95], chapter 14, pages 330–349. Download at `http://www.acsac.org/secshelf/book001/book001.html`.

[APS99] V. Apostolopoulos, V. Peris, and D. Saha. Transport layer security: How much does it really cost? In *Conference on Computer Communications (IEEE Infocom)*, pages 717–725. IEEE Computer Society, New York, March 1999.

[AR00a] M. Abadi and P. Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). In J. van Leeuwen, O. Watanabe, M. Hagiya, D. Mosses P. and T. Ito, editors, *Theoretical Computer Science, International Conference IFIP TCS 2000*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer, Berlin Heidelberg New York, 2000.

[AR00b] E. Astesiano and G. Reggio. Formalism and method. *Theoretical Computer Science*, 236:3–34, 2000.

[Arc01] ArcSecure. `http://www.arcsecure.de`, February 2001.

[AS85] B. Alpern and F. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, October 1985.

[AW03] K. Alghathbar and D. Wijesekera. Consistent and complete access control policies in use cases. In Stevens [Ste03b], pages 373–387.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society, Series A*, 426(1871):233–271, December 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in *ACM Transactions on Computer Systems 8*, 1:18–36 (February 1990).

[BB03] E. Boiten and M. Bujorianu. Exploring UML development through unification. In Jürjens et al [JRFF03], pages 47–62.

[BBD+03] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. R. Nielson. Automatic validation of protocol narration. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003)*, pages 126–140. IEEE Computer Society, New York, 2003.

[BBH+03] R. Breu, K. Burger, M. Hafner, J. Jürjens, G. Popp, G. Wimmel, and V. Lotz. Key issues of a formally based process model for security engineering. In *16th International Conference "Software & Systems Engineering & their Applications" (ICSSEA 2003)*, 2003.

[BCR00] E. Börger, A. Cavarra, and E. Riccobene. Modeling the dynamics of UML State Machines. In Y. Gurevich, P. Kutter, M. Odersky, and L. Thiele, editors, *Abstract State Machines. Theory and Applications*, volume 1912 of *Lecture Notes in Computer Science*, pages 223–241. Springer, Berlin Heidelberg New York, 2000.

[BCR04] E. Börger, A. Cavarra, and E. Riccobene. On formalizing UML state machines using ASM. *Information & Software Technology*, 46(5):287–292, 2004.

[BD00] C. Bolton and J. Davies. Using relational and behavioural semantics in the verification of object models. In Smith and Talcott [ST00], pages 163–182.

[BDCL+01] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia. Dependability analysis in the early phases of UML based system design. *Journal of Computer Systems Science and Engineering*, 16:265–275, 2001.

[BDL03] D. Basin, J. Doser, and T. Lodderstedt. Model driven security for process-oriented systems. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, pages 100–109. ACM, New York, 2003.

[BDNN01] C. Bodei, P. Degano, F. Nielson, and H. R. Nielson. Security analysis using flow logics. In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*. World Scientific, 2001.

[BdVS02] P. Bonatti, S. De Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, February 2002.

[Bel98] S. M. Bellovin. Cryptography and the internet. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes*

in Computer Science, pages 46–55. Springer, Berlin Heidelberg New York, 1998.

[BFPR02] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Transforming processes to check and ensure information flow security. In H. Kirchner and C. Ringeissen, editors, *9th International Conference on Algebraic Methodology and Software Technology (AMAST 2002)*, volume 2422 of *Lecture Notes in Computer Science*, pages 271–286. Springer, Berlin Heidelberg New York, 2002.

[BG03] J. Barros and L. Gomes. Actions as activities and activities as Petri nets. In Jürjens et al [JRFF03], pages 129–135.

[BGH+97] R. Breu, R. Grosu, F. Huber, B. Rumpe, and W. Schwerin. Towards a precise semantics for object-oriented modeling techniques. In J. Bosch and S. Mitchell, editors, *Object-Oriented Technology, ECOOP 1997 Workshop Reader*, volume 1357 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 1997.

[BGH+98] R. Breu, R. Grosu, F. Huber, B. Rumpe, and W. Schwerin. Systems, views and models of UML. In M. Schader and A. Korthaus, editors, *The Unified Modeling Language – Technical Aspects and Applications*, pages 93–109. Physica-Verlag, Heidelberg, 1998. 1st UML Workshop of the GROOM GI-Arbeitskreis.

[BHH+97] R. Breu, U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe, and V. Thurner. Towards a formalization of the unified modeling language. In M. Aksit and S. Matsuoka, editors, *ECOOP'97 - Object-Oriented Programming*, volume 1241 of *Lecture Notes in Computer Science*, pages 344–366. Springer, Berlin Heidelberg New York, 1997.

[BHP+97] M. Broy, F. Huber, B. Paech, B. Rumpe, and K. Spies. Software and system modeling based on a unified formal semantics. In M. Broy and B. Rumpe, editors, *Requirements Targeting Software and Systems Engineering (RTSE '97)*, volume 1526 of *Lecture Notes in Computer Science*, pages 43–68. Springer, Berlin Heidelberg New York, 1997.

[BJMF02] M. Boger, M. Jeckle, S. Mueller, and J. Fransson. Diagram Interchange for UML. In Jézéquel et al [JHC02], pages 398–411.

[BKL02] G. Brose, M. Koch, and K.-P. Löhr. Integrating access control design into the software development process. In *Integrated Design and Process Technology (IDPT)*, 2002.

[Bla04] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *2004 IEEE Symposium on Security and Privacy*, pages 86–100. IEEE Computer Society, New York, 2004.

[Blo02] B. Blobel. Aspects of modeling using the examples of Electronic Health Records (EHRs). In *CORAS workshop*, 2002. Part of International Conference on Telemedicine (ICT2002).

[BMM99] S. Brackin, C. Meadows, and J. Millen. CAPSL interface for the NRL protocol analyzer. In *Application-Specific Systems and Software Engineering Technology (ASSET 1999)*. IEEE Computer Society, New York, 1999.

[BMV03] D. A. Basin, S. Mödersheim, and Luca Viganò. An on-the-fly model-checker for security protocol analysis. In Snekkenes and Gollmann [SG03], pages 253–270.

[Boe81] B.W. Boehm. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, NJ, 1981.

[Boo91] G. Booch. *Object-Oriented Design With Applications*. Benjamin/Cummings, Redwood City, CA, 1991.

[BP04] R. Breu and G. Popp. Actor-centric modeling of user rights. In Wermelinger and Margaria [WM04], pages 165–179.

[BR97] G. Bella and E. Riccobene. Formal analysis of the Kerberos authentication system. *Journal of Universal Computer Science*, 3(12):1337–1381, December 1997.

[BR98] G. Bella and E. Riccobene. A realistic environment for crypto-protocol analyses by ASMs. In U. Glasser, editor, *5th International Workshop on Abstract State Machines (ASM)*, pages 127–138, 1998. Part of INFORMATIK 1998.

[Bre01] R. Breu. *Objektorientierter Softwareentwurf - Integration mit UML*. Springer, Berlin Heidelberg New York, 2001.

[Bro86] M. Broy. A theory for nondeterminism, parallelism, communication, and concurrency. *Theoretical Computer Science*, 45:1–61, 1986.

[Bro97] M. Broy. Compositional refinement of interactive systems. *Journal of the ACM*, 44(6):850–891, 1997.

[Bro98] M. Broy. A functional rephrasing of the assumption/commitment specification style. *Formal Methods in System Design*, 13(1):87–119, 1998.

[Bro00] M. Broy. Algebraic specification of reactive systems. *Theoretical Computer Science*, 239(1):3–40, 2000.

[Bro01] M. Broy. Toward a mathematical foundation of software engineering methods. *IEEE Transactions on Software Engineering*, 27(1):42–57, 2001.

[BS00] E. Börger and J. Schmid. Composition and submachine concepts for sequential ASMs. In P. Clote and H. Schwichtenberg, editors, *Computer Science Logic (CSL 2000)*, volume 1862 of *Lecture Notes in Computer Science*, pages 41–60. Springer, Berlin Heidelberg New York, 2000.

[BS01] M. Broy and K. Stølen. *Specification and Development of Interactive Systems*. Springer, Berlin Heidelberg New York, 2001.

[BS03] M. Broy and R. Steinbrüggen. *Modellbildung in der Informatik*. Springer, Berlin Heidelberg New York, 2003.

[BV99] B. Bokowski and J. Vitek. Confined types. In *14th Annual ACM SIGPLAN Conference on ObjectOriented Programming Systems, Languages, and Applications (OOPSLA 1999)*, pages 82–96, 1999.

[BW00] M. Broy and M. Wirsing. Algebraic state machines. In T. Rus, editor, *8th International Conference on Algebraic Methodology and Software Technology (AMAST 2000)*, volume 1816 of *Lecture Notes in Computer Science*, pages 89–188. Springer, Berlin Heidelberg New York, 2000.

[Cas03] Castor. Castor library. Available at `http://castor.exolab.org`, June 2003.

[Cav00] A. Cavarra. *Applying Abstract State Machines to Formalize and Integrate the UML Lightweight Method*. PhD thesis, DMI, Universitá di Catania, 2000.

[CC01] Common Criteria. `http://csec.nist.gov/cc/`, 2001.

[CCG+03] S. Chaki, E. M. Clarke, A. Groce, S. Jha, and H. Veith. Modular verification of software components in C. In *25th International Conference on Software Engineering (ICSE 2003)*, pages 385–395. IEEE Computer Society, New York, 2003.

[CEP01] CEPSCO. Common Electronic Purse Specifications, 2001. Business Requirements Version 7.0, Functional Requirements Version 6.3, Technical Specification Version 2.3, available from `http://www.cepsco.com`.

[CER] CERT Coordination Center (CERT/CC) `http://www.cert.org`.

[Chu93] L. Chung. Dealing with security requirements during the development of information systems. In *5th International Conference on Advanced Information Systems Engineering (CAiSE 1993)*, pages 234–251. Springer, Berlin Heidelberg New York, 1993.

[CKM+99] S. Cook, A. Kleppe, R. Mitchell, B. Rumpe, J. Warmer, and A. Wills. Defining UML family members using prefaces. In Ch. Mingins and B. Meyer, editors, *Proceedings of Technology of Object-Oriented Languages and Systems (TOOLS Pacific 1999)*. IEEE Computer Society, New York, 1999.

[CLM+03] S. Cigoli, P. Leblanc, S. Malaponti, D. Mandrioli, M. Mazzucchelli, and A. Morzenti. An experiment in applying UML 2.0 to the development of an industrial critical application. In Jürjens et al [JRFF03], pages 19–34.

[Coo00] S. Cook. The UML family: Profiles, prefaces and packages. In Evans et al [EKS00], pages 255–264.

[CRS04] A. Cavarra, E. Riccobene, and P. Scandurra. A framework to simulate uml models: moving from a semi-formal to a formal environment. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, *2004 ACM Symposium on Applied Computing (SAC)*, pages 1519–1523, 2004.

[CW96] E. Clarke and J. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

[DB00] J. Derrick and E. Boiten. Refinement of objects and operations in Object-Z. In Smith and Talcott [ST00], pages 257–277.

[DB01] J. Derrick and E. Boiten. *Refinement in Z and Object-Z: Foundations and advanced applications*. Formal Approaches to Computing and Information Technology. Springer, Berlin Heidelberg New York, 2001.

[DBG01] J. Dushina, M. Benjamin, and D. Geist. Semi-Formal Test Generation with Genevieve. In *38th Design Automation Conference (DAC)*, pages 617–622. ACM, New York, 2001. Download at `http://www.dac.com`.

[DEG01] Degas, 2001. `http://www.omnys.it/degas`.

[DF93] J. Dick and A. Faivre. Automating the generation and sequencing of test cases from model-based specifications. In *Formal Methods Europe (FME) 1993: Industrial-Strength Formal Methods*, volume 670 of *Lecture Notes in Computer Science*, pages 268–284. Springer, Berlin Heidelberg New York, 1993.

[DFG00] A. Durante, R. Focardi, and R. Gorrieri. A compiler for analyzing cryptographic protocols using noninterference. *ACM Transactions on Software Engineering and Methodology*, 9(4):488–528, 2000.

[DFS98] P. Devanbu, P. Fong, and S. Stubblebine. Techniques for trusted software engineering. In *20th International Conference on Software Engineering*, pages 126–135, 1998.

[DGH02] K. Diethers, U. Goltz, and M. Huhn. Model checking UML statecharts with time. In Jürjens et al [JCF+02], pages 35–52.

[DHS03] A. Darvas, R. Hähnle, and D. Sands. A theorem proving approach to analysis of secure information flow. In Gorrieri [Gor03], pages 111–120. Available at `http://www.dsi.unive.it/IFIPWG1_7/WITS2003/program-wits03.htm`.

[Dij68] E. W. Dijkstra. Stepwise program construction. published as [Dij82], February 1968.

[Dij82] E. W. Dijkstra. Stepwise program construction. In *Selected Writings on Computing: A Personal Perspective*, pages 1–14. Springer, Berlin Heidelberg New York, 1982.

[DRR+02] T. Dimitrakos, B. Ritchie, D. Raptis, J. Ø. Aagedal, F. den Braber, K. Stølen, and S. H. Houmb. Integrating model-based security risk management into ebusiness systems development: The CORAS approach. In J. Monteiro, P. Swatman, and L. Tavares, editors, *Second IFIP Conference on E-Commerce, E-Business, E-Government (I3E 2002)*, volume 233 of *IFIP Conference Proceedings*, pages 159–175. Kluwer Academic, Dordrecht, 2002.

[DS00a] J. Derrick and G. Smith. Structural refinement in Object-Z/CSP. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *Integrated Formal Methods (IFM 2000)*, volume 1945 of *Lecture Notes in Computer Science*, pages 194–213. Springer, Berlin Heidelberg New York, 2000.

[DS00b] P. Devanbu and S. Stubblebine. Software engineering for security: A roadmap. In *The Future of Software Engineering*, pages 227–239, 2000. Special Volume of the International Conference on Software Engineering (ICSE 2000).

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2):198–208, 1983.

[Eck95] C. Eckert. Matching security policies to application needs. In J. H. P. Eloff and S. H. von Solms, editors, *11th International Conference on Information Security (SEC 1995)*, pages 237–254. Chapman & Hall, London, 1995.

[Eck98] C. Eckert. Tool-supported verification of cryptographic protocols. In *14th International Information Security Conference (SEC 1998)*. Chapman & Hall, London, 1998.

[Eck03] C. Eckert. *IT-Sicherheit – Konzepte, Verfahren, Protokolle*. Oldenbourg, München, 2nd edition, 2003.

[EFLR99] A. Evans, R. B. France, K. Lano, and B. Rumpe. Meta-modeling semantics of UML. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Behavioral Specifications of Businesses and Systems*. Kluwer Academic, Amsterdam, 1999.

[eHe03] ehealth project webpages. `http://www.dimdi.de/de/ehealth/karte`, 2003.

[EHHS00] G. Engels, J. Hausmann, R. Heckel, and S. Sauer. Dynamic metamodeling: A graphical approach to the operational semantics of behavioral diagrams in UML. In Evans et al [EKS00], pages 323–337.

[EKS00] A. Evans, S. Kent, and B. Selic, editors. *The Unified Modeling Language: Advancing the Standard (UML 2000)*, volume 1939 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2000.

[EM97] C. Eckert and D. Marek. Developing secure applications: A systematic approach. In *13th International Conference on Information Security (SEC 1998)*, pages 267–279. Chapman & Hall, London, 1997.

[ERW81] B. Fernandez E. C. Summers R. and C. Wood. *Database Security and Integrity*. Systems Programming Series. Addison-Wesley, Reading, MA, February 1981.

[FELR98] R. B. France, A. Evans, K. Lano, and B. Rumpe. The UML as a formal modeling notation. *Computer Standards & Interfaces*, 19:325–334, 1998.

[Fer98] E. B. Fernandez. A holistic view of internet security, 1998. Tutorial notes.

[Fer99] E. B. Fernandez. Coordination of security levels for internet architectures. In *International Workshop on Database and Expert Systems Applications (DEXA 1999)*, pages 837–841, 1999.

[FG95] R. Focardi and R. Gorrieri. A taxonomy of security properties for process algebras. *Journal of Computer Security*, 3(1):5–34, 1995.

[FG97] R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transaction of Software Engineering*, 23(9):550–571, September 1997.

[FGG97] R. Focardi, A. Ghelli, and R. Gorrieri. Using non interference for the analysis of security protocols. In H. Orman and C. Meadows, editors, *DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997. Available at `http://dimacs.rutgers.edu/Workshops/Security/program2/program.html`.

[FGM03] R. Focardi, R. Gorrieri, and F. Martinelli. A comparison of three authentication properties. *Theoretical Computer Science*, 291(3):285–327, 2003.

[FH97] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *Workshop on Role-Based Access Control*, pages 121–125. ACM, New York, 1997.

[FHH+03] R. B. France, H. Hermanns, H. Hußmann, A. Knapp, B. Selic, and J. Jürjens. What's wrong with uml for critical systems design ? In Jürjens et al [JRFF03]. Panel session.

[FJ02] E. B. Fernandez and J. Jürjens. A holistic view of secure systems development: Using patterns and UML. In *17th International Conference on Information Security (SEC 2002)*. International Federation for Information Processing (IFIP), 2002. Half-day tutorial.

[FKG+02] R. Fredriksen, M. Kristiansen, B.A. Gran, K. Stølen, T.A. Opperud, and T. Dimitrakos. The CORAS framework for a model-based risk management process. In S. Anderson, S. Bologna, and M. Felici, editors, *21st International Conference on Computer Safety, Reliabiltiy and Security (SAFECOMP 2002)*, volume 2434 of *Lecture Notes in Computer Science*, pages 94–105. Springer, Berlin Heidelberg New York, 2002.

[FMMMP02] E. Fernández-Medina, A. Martínez, C. Medina, and M. Piattini. UML for the design of secure databases: Integrating security levels, user roles, and constraints in the database design process. In Jürjens et al [JCF+02], pages 93–106.

[Fow04] M. Fowler. *UML Distilled*. Addison-Wesley, Reading, MA, third edition, 2004.

[FP01] E. B. Fernandez and R. Y. Pan. A pattern language for security models. In *Conference on Pattern Languages of Programs (PLoP 2001)*, 2001. `http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/accepted-papers.html`.

[FPR00] M. Fontura, W. Pree, and B. Rumpe. UML-F: A modeling language for object-oriented frameworks. In E. Bertino, editor, *14th European Conference on Object-Oriented Programming (ECOOP 2000)*, volume 1850 of *Lecture Notes in Computer Science*, pages 63–82. Springer, Berlin Heidelberg New York, 2000.

[Gas88] M. Gasser. *Building a secure computer system.* Van Nostrand Reinhold, New York, 1988.

[GB99] S. Goldwasser and M. Bellare. Lecture notes on cryptography. Summer Course "Cryptography and Computer Security" at MIT, 1999.

[GCS91] J. Graham-Cumming and J. Sanders. On the refinement of noninterference. In *Computer Security Foundations Workshop (CSFW)*, pages 35–42, 1991.

[Gen03] Gentleware. `http://www.gentleware.com`, 2003.

[GFR02] G. Georg, R. B. France, and I. Ray. An aspect-based approach to modeling security concerns. In Jürjens et al [JCF+02], pages 107–120.

[GFR03] G. Georg, R. B. France, and I. Ray. Creating security mechanism aspect models from abstract security aspect models. In Jürjens et al [JRFF03], pages 35–46.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA, 1995.

[GHJW03] J. Grünbauer, H. Hollmann, J. Jürjens, and G. Wimmel. Modelling and verification of layered security protocols: A bank application. In *Computer Safety, Reliability, and Security (SAFECOMP 2003)*, volume 2788 of *Lecture Notes in Computer Science*, pages 116–129. Springer, Berlin Heidelberg New York, 2003.

[GHR03] J. D. Guttman, A. L. Herzog, and J. D. Ramsdell. Information flow in operating systems: Eager formal methods. In Gorrieri [Gor03], pages 81–90. Available at `http://www.dsi.unive.it/IFIPWG1_7/WITS2003/program-wits03.htm`.

[GI98] Gesellschaft für Informatik. *28th Annual Conference of the German Society of Computer Science.* Technical Report, Magdeburg University, 1998.

[GKW02] D. Gollmann, G. Karjoth, and M. Waidner, editors. *7th European Symposium on Research in Computer Security (ESORICS 2002)*, volume 2502 of *Lecture Notes in Computer Science.* Springer, Berlin Heidelberg New York, 2002.

[GL97] F. Germeau and G. Leduc. Model-based design and verification of security protocols using LOTOS. In H. Orman and C. Meadows, editors, *DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997. Available at `http://dimacs.rutgers.edu/Workshops/Security/program2/program.html`.

[GM82] J. Goguen and J. Meseguer. Security policies and security models. In *Symposium on Security and Privacy (S&P)*, pages 11–20. IEEE Computer Society, New York, 1982.

[GM84] J. Goguen and J. Meseguer. Unwinding and inference control. In *Symposium on Security and Privacy (S&P)*, pages 75–87. IEEE Computer Society, New York, 1984.

[GMM03] P. Giorgini, F. Massacci, and J. Mylopoulos. Requirement engineering meets security: A case study on modelling secure electronic transactions by VISA and Mastercard. In I.-Y. Song, S. W. Liddle, T. W. Ling, and P. Scheuermann, editors, *22nd International Conference on Conceptual Modeling (ER 2003)*, volume 2813 of *Lecture Notes in Computer Science*, pages 263–276. Springer, Berlin Heidelberg New York, 2003.

[GNY90] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Symposium on Security and Privacy (S&P)*, pages 234–248. IEEE Computer Society, New York, 1990.

[Gol96] D. Gollmann. What do we mean by entity authentication? In *Symposium on Security and Privacy (S&P)*, pages 46–54, 1996.

[Gol99] D. Gollmann. *Computer Security.* John Wiley & Sons, New York, 1999.

[Gol00] D. Gollmann. On the verification of cryptographic protocols – a tale of two committees. In S. Schneider and P. Ryan, editors, *Workshop on Security Architectures and Information Flow*, volume 32 of *Electronic Notes in Theoretical Computer Science*. Elsevier, Amsterdam, 2000.

[Gol03a] D. Gollmann. Analysing security protocols. In A. Abdallah, P. Ryan, and S. Schneider, editors, *Formal Aspects of Security (FASec 2002)*, volume 2629 of *Lecture Notes in Computer Science*, pages 71–80. Springer, Berlin Heidelberg New York, 2003.

[Gol03b] D. Gollmann. Authentication by correspondence. *IEEE Journal on Selected Areas in Communications*, 21(1):88–95, January 2003.

[Gol03c] D. Gollmann. Facets of security. In C. Priami, editor, *Global Computing. Programming Environments, Languages, Security, and Analysis of Systems, IST/FET International Workshop, (GC 2003)*, volume 2874 of *Lecture Notes in Computer Science*, pages 192–202. Springer, Berlin Heidelberg New York, 2003.

[Gon98] L. Gong. Java^{TM} Security Architecture (JDK1.2). `http://java.sun.com/j2se/1.4.2/docs/guide/security/spec/security-spec.doc.html`, October 1998.

[Gon99] L. Gong. *Inside Java 2 Platform Security – Architecture, API Design, and Implementation.* Addison-Wesley, Reading, MA, 1999.

[GOR02] S. Gürgens, P. Ochsenschläger, and C. Rudolph. Role based specification and security analysis of cryptographic protocols using asynchronous product automata. In *DEXA Workshops 2002*, pages 473–482. IEEE Computer Society, New York, 2002.

[Gor03] R. Gorrieri, editor. *Workshop on Issues in the Theory of Security (WITS'03).* IFIP WG 1.7, ACM SIGPLAN, and GI FoMSESS, 2003. Available at `http://www.dsi.unive.it/IFIPWG1_7/WITS2003/program-wits03.htm`.

[GPP98] M. Gogolla and F. Parisi-Presicce. State diagrams in UML: A formal semantics using graph transformations. In M. Broy, D. Coleman, T. Maibaum, and B. Rumpe, editors, *Workshop on Precise Semantics for Software Modeling Techniques (PSMT 1998)*, pages 55–72. TU München, TUM-I9803, 1998.

[GSG99] S. Gritzalis, D. Spinellis, and P. Georgiadis. Security protocols over open networks and distributed systems: Formal methods for their analysis, design, and verification. *Computer Communications Journal*, 22(8):695–707, 1999.

[GSS00] W. Grieskamp, T. Santen, and B. Stoddart, editors. *2nd International Conference on Integrated Formal Methods (IFM 2000)*, volume 1945 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2000.

[Gur95] Y. Gurevich. Evolving algebras 1993: Lipari guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, Oxford, 1995.

[Gur97] Y. Gurevich. Draft of the ASM Guide. Technical Report CSE-TR-336-97, University of Michigan, EECS Department, May 1997.

[Gut00] J. Guttman. Security goals: Packet trajectories and strand spaces. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design (FOSAD 2000)*, volume 2171 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2000. Summer school lectures.

[HBVL97] T. Hillenbrand, A. Buch, R. Vogt, and B. Lochner. WALDMEISTER – high-performance equational deduction. *Journal of Automated Reasoning*, 2(18):265–270, 1997.

[HdBLS02] S. H. Houmb, F. den Braber, M. S. Lund, and K. Stølen. Towards a UML profile for model-based risk assessment. In Jürjens et al [JCF+02], pages 79–92.

[Hei99] C. Heitmeyer. Formal methods for developing software specifications: Paths to wider usage. In H. R. Arabnia, editor, *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 1999)*, pages 1047–1053, 1999.

[Hei01] C. Heitmeyer. Applying "practical" formal methods to the specification and analysis of security properties. In V. I. Gorodetski, V. A. Skormin, and L. J. Popyack, editors, *International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS 2001)*, volume 2052 of *Lecture Notes in Computer Science*, pages 84–89. Springer, Berlin Heidelberg New York, 2001.

[Her03] P. Herrmann. Formal security policy verification of distributed component-structured software. In H. König, M. Heiner, and A. Wolisz, editors, *23rd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2003)*, volume 2767 of *Lecture Notes in Computer Science*, pages 257–272. Springer, Berlin Heidelberg New York, 2003.

[HF97] J. C. Hawkins and E. B. Fernandez. Extending use cases and interaction diagrams to develop distributed system architecture requirements. Technical Report TR-CSE-97-47, Department of Computer Science & Engineering, Florida Atlantic University, 1997.

[HG97] D. Harel and E. Gery. Executable object modeling with statecharts. *Computer*, 30(7):31–42, 1997.

[HG02] K. Hansen and I. Gullesen. Utilizing UML and patterns for safety critical systems. In Jürjens et al [JCF+02], pages 147–154.

[HH03a] R. Heldal and F. Hultin. Bridging model-based and language-based security. In Snekkenes and Gollmann [SG03], pages 235–252.

[HH03b] S. H. Houmb and K. Hansen. Towards a UML profile for model-based risk assessment of security critical systems. In Jürjens et al [JRFF03], pages 95–104.

[Hit01] R. Hite. Oral communication, May 2001.

[HJ03a] S. Höhn and J. Jürjens. Automated checking of SAP security permissions. In *6th Working Conference on Integrity and Internal Control in Information Systems (IICIS)*. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2003.

[HJ03b] S. H. Houmb and J. Jürjens. Developing secure networked web-based systems using model-based risk assessment and UMLsec. In *10th Asia-Pacific Software Engineering Conference (APSEC 2003)*, page 488ff. IEEE Computer Society, New York, 2003.

[HK98] M. Hitz and G. Kappel. Developing with UML – goodies, pitfalls, workarounds. In J. Bézivin and P.-A. Muller, editors, *UML 1998 – Beyond the Notation*, volume 1618 of *Lecture Notes in Computer Science*, pages 9–20. Springer, Berlin Heidelberg New York, 1998.

[HMR+98] F. Huber, S. Molterer, A. Rausch, B. Schätz, M. Sihling, and O. Slotosch. Tool supported specification and simulation of distributed systems. In *International Symposium on Software Engineering for Parallel and Distributed Systems*, pages 155–164, 1998.

[HNS97] S. Helke, T. Neustupny, and T. Santen. Automating test case generation from Z specifications with Isabelle. In J. Bowen, M. Hinchey, and D. Till, editors, *Proceedings of the Z Users Conference (ZUM 1997): The Z Formal Specification Notation*, volume 1212 of *Lecture Notes in Computer Science*, pages 52–71. Springer, Berlin Heidelberg New York, 1997.

[Hoa72] C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–282, 1972.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, Englewood Cliffs, NJ, 1985.

[Hoa96] C. A. R. Hoare. How did software get so reliable without proof? In M.-C. Gaudel and J. Woodcock, editors, *Formal Methods Europe 1996 (FME): Industrial Benefit and Advances in Formal Methods*, volume 1051 of *Lecture Notes in Computer Science*, pages 1–17. Springer, Berlin Heidelberg New York, 1996.

[Hol03] G. Holzmann. *The Spin Model Checker*. Addison-Wesley, 2003.

[HPS01] M. Heisel, A. Pfitzmann, and T. Santen. Confidentiality-preserving refinement. In *Computer Security Foundations Workshop (CSFW)*, pages 295–306. IEEE Computer Society, New York, 2001.

[HTB03] R. Hawkins, I. Toyn, and I. Bate. An approach to designing safety critical systems using the Unified Modelling Language. In Jürjens et al [JRFF03], pages 3–18.

[Huß01] H. Hußmann, editor. *4th International Conference on Fundamental Approaches to Software Engineering (FASE)*, volume 2029 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2001.

[Jan02] D. Jansen. Probabilistic UML statecharts for specification and verification: A case study. In Jürjens et al [JCF+02], pages 121–132.

[JBR98] I. Jacobsen, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, Reading, MA, 1998.

[JCF+02] J. Jürjens, V. Cengarle, E. B. Fernandez, B. Rumpe, and R. Sandner, editors. *Critical Systems Development with UML (CSDUML 2002)*, TU München Technical Report TUM-I0208, 2002. UML 2002 satellite workshop proceedings.

[JFS04] J. Jürjens, E. B. Fernandez, and R. Sandner. Critical systems development with UML-like languages. *Journal on Software and Systems Modeling*, 2004. Special section, to be published.

[JG03] J. Jürjens and J. Grünbauer. Critical systems development with UML: Overview with automotive case-study. In *4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2003)*, pages 512–517. International Association for Computer and Information Science (ACIS), 2003.

[JH03] J. Jürjens and S. H. Houmb. Development of safety-critical systems and model-based risk analysis with UML. In *Dependable Computing*, volume 2847 of *Lecture Notes in Computer Science*, pages 364–365. Springer, Berlin Heidelberg New York, 2003.

[JH04] J. Jürjens and S. K. Houmb. Risk-driven development of security-critical systems using UMLsec. Kluwer Academic, Dordrecht, 2004. Book chapter to be published (book title pending).

[JHC02] J.-M. Jézéquel, H. Hußmann, and S. Cook, editors. *5th International Conference on the Unified Modeling Language (UML 2002)*, volume 2460 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2002.

[JHK03] D. N. Jansen, H. Hermanns, and J.-P. Katoen. A QoS-oriented extension of UML statecharts. In Stevens [Ste03b], pages 76–91. 6th International Conference.

[JK03] J. Jürjens and T. Kuhn. Mobile web-based applications with UML: Concepts and tools. In *International Conference WWW/Internet 2003*. International Association for Development of the Information Society (IADIS), 2003. Half-day tutorial.

[JKS95] S. Jajodia, B. Kogan, and R. Sandhu. A multilevel-secure object-oriented data model. In Abrams et al [AJP95]. Download at `http://www.acsac.org/secshelf/book001/book001.html`.

[Jon72] C. B. Jones. Formal development of correct algorithms: An example based on Earley's recogniser. In *ACM Conference on Proving Assertions about Programs*, volume 7 of *SIGPLAN Notices*, pages 150–169, 1972.

[Jon87] B. Jonsson. *Compositional Verification of Distributed Systems*. PhD thesis, Department of Computer Systems, Uppsala University, 1987. Technical Report DoCS 87/09.

[JPW02] J. Jürjens, G. Popp, and G. Wimmel. Towards using security patterns in model-based system development. In *7th European Conference on Pattern Languages of Programs (EuroPLoP 2002)*, 2002. Security Focus Group.

[JPW03] J. Jürjens, G. Popp, and G. Wimmel. Use case oriented development of security-critical systems. In *Workshop on Distributed Objects*

and Components Security (DOCsec 2003). Object Management Group (OMG), 2003. Also appeared in *Information Security Bulletin* 8:51–56, 2003.

[JRFF03] J. Jürjens, B. Rumpe, R. B. France, and E. B. Fernandez, editors. *Critical Systems Development with UML (CSDUML 2003)*, number TUM-I0317 in TU München Technical Report, 2003. UML 2003 satellite workshop proceedings.

[JS04a] J. Jürjens and P. Shabalin. Automated verification of UMLsec models for security requirements. In J.-M. Jézéquel, H. Hußmann, and S. Cook, editors, *UML 2004 – The Unified Modeling Language*, volume 2460 of *Lecture Notes in Computer Science*, pages 412–425. Springer, Berlin Heidelberg New York, 2004.

[JS04b] J. Jürjens and P. Shabalin. XML-based analysis of UML models for critical systems development. In *Advances in UML and XML Based Software Evolution*. IDEA Group Publishing, 2004. To be published.

[JS04c] J. Jürjens and Pasha Shabalin. A foundation for tool-supported critical systems development with UML. In *11th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS 2004)*, Brno, Czech Republic, May 24–26 2004. IEEE Computer Society, New York.

[JSA+04] J. Jürjens, P. Shabalin, E. Alter, A. Gilg, S. Höhn, D. Kopjev, M. Lehrhuber, S. Meng, M. Schwaiger, G. Kokavecz, S. Schwarzmüller, and S. Shen. UMLsec tool, 2004. Accessible through a webinterface via [Jür04]. Available as open-source.

[Jür00] J. Jürjens. Secure information flow for concurrent processes. In C. Palamidessi, editor, *CONCUR 2000 (11th International Conference on Concurrency Theory)*, volume 1877 of *Lecture Notes in Computer Science*, pages 395–409. Springer, Berlin Heidelberg New York, 2000.

[Jür01a] J. Jürjens. Abstracting from failure probabilities. In *Second International Conference on Application of Concurrency to System Design (ACSD 2001)*, pages 53–64. IEEE Computer Society, New York, 2001.

[Jür01b] J. Jürjens. Composability of secrecy. In V. Gorodetski, V. Skormin, and L. Popyack, editors, *International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS 2001)*, volume 2052 of *Lecture Notes in Computer Science*, pages 28–38. Springer, Berlin Heidelberg New York, 2001.

[Jür01c] J. Jürjens. Developing secure systems with UMLsec – from business processes to implementation. In D. Fox, M. Köhntopp, and A. Pfitzmann, editors, *Verläßliche IT-Systeme 2001 (VIS 2001)*, DuD-Fachbeiträge. Vieweg, Wiesbaden, 2001.

[Jür01d] J. Jürjens. Formal development and verification of security-critical systems with UML. In *Workshop on Automated Verification of Critical Systems (AVoCS 2001)*, Oxford, 2001. Published as OUCL Technical Report PRG-RR-01-07. Available at `ftp://ftp.comlab.ox.ac.uk/pub/Documents/techreports/RR-01-07.ps.gz`.

[Jür01e] J. Jürjens. Modelling audit security for smart-card payment schemes with UMLsec. In M. Dupuy and P. Paradinas, editors, *Trusted Information: The New Decade Challenge*, pages 93–108. International Federation for Information Processing (IFIP), Kluwer Academic, Dor-

drecht, 2001. Proceedings of the *16th International Conference on Information Security (SEC 2001)*.

[Jür01f] J. Jürjens. On a problem of Gabriel and Ulmer. *Journal of Pure and Applied Algebra*, 158:183–196, 2001.

[Jür01g] J. Jürjens. Secrecy-preserving refinement. In *International Symposium on Formal Methods Europe (FME)*, volume 2021 of *Lecture Notes in Computer Science*, pages 135–152. Springer, Berlin Heidelberg New York, 2001.

[Jür01h] J. Jürjens. Secure Java development with UMLsec. In B. De Decker, F. Piessens, J. Smits, and E. Van Herrenweghen, editors, *Advances in Network and Distributed Systems Security*, pages 107–124. International Federation for Information Processing (IFIP) TC-11 WG 11.4, Kluwer Academic, Dordrecht, 2001. Proceedings of the *First Annual Working Conference on Network Security (I-NetSec 2001)*.

[Jür01i] J. Jürjens. Towards development of secure systems using UMLsec. In H. Hußmann, editor, *4th International Conference on Fundamental Approaches to Software Engineering (FASE)*, volume 2029 of *Lecture Notes in Computer Science*, pages 187–200. Springer, Berlin Heidelberg New York, 2001. Also Oxford University Computing Laboratory TR-9-00 (November 2000), `http://web.comlab.ox.ac.uk/oucl/publications/tr/tr-9-00.html`.

[Jür01j] J. Jürjens. Transformations for introducing patterns – a secure systems case study. In *Workshop on Transformations in UML (WTUML, ETAPS 2001 Satellite Event)*, 2001.

[Jür02a] J. Jürjens. A UML statecharts semantics with message-passing. In Lamont et al [LHPP02], pages 1009–1013.

[Jür02b] J. Jürjens. Critical systems development with UML. In *Forum on Design Languages (FDL 2002)*. European Electronic Chips & Systems design Initiative (ECSI), 2002. Invited talk.

[Jür02c] J. Jürjens. Encapsulating rules of prudent security engineering. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *Security Protocols*, volume 2467 of *Lecture Notes in Computer Science*, pages 95–101. Springer, Berlin Heidelberg New York, 2002. 9th International Workshop. Transcript of discussion on pages 102–106.

[Jür02d] J. Jürjens. Formal semantics for interacting UML subsystems. In B. Jacobs and A. Rensink, editors, *5th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS 2002)*, pages 29–44. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2002.

[Jür02e] J. Jürjens. Methodische Entwicklung sicherer CORBA-Anwendungen. In P. Horster, editor, *Enterprise Security*. IT-Verlag, Sauerlach, 2002.

[Jür02f] J. Jürjens. *Principles for Secure Systems Design*. PhD thesis, Oxford University Computing Laboratory, 2002.

[Jür02g] J. Jürjens. Secure systems development with UML – applications to telemedicine. In *CORAS Public Workshop*, 2002. International Conference on Telemedicine (ICT 2002). Invited talk.

[Jür02h] J. Jürjens. UMLsec – presenting the profile. In *6th Annual Workshop on Distributed Objects and Components Security (DOCsec 2002)*. Object Management Group (OMG), 2002. Half-day tutorial.

[Jür02i] J. Jürjens. UMLsec: Extending UML for secure systems development. In Jézéquel et al [JHC02], pages 412–425.

[Jür02j] J. Jürjens. Using UMLsec and goal-trees for secure systems development. In Lamont et al [LHPP02], pages 1026–1031.

[Jür03a] J. Jürjens. Algebraic state machines: Concepts and applications to security. In M. Broy and A. Zamulin, editors, *Andrei Ershov 5th International Conference "Perspectives of System Informatics" (PSI 2003)*, volume 2890 of *Lecture Notes in Computer Science*, pages 338–343. Springer, Berlin Heidelberg New York, 2003.

[Jür03b] J. Jürjens. Critical systems development with UML and model-based testing, 2003. Tutorials at SAFECOMP 2003, ICSTEST-E 2003, SE 2004. Download of material at `http://www4.in.tum.de/~juerjens/csdumltut`.

[Jür03c] J. Jürjens. Developing safety- and security-critical systems with UML. In *DARP workshop*, 2003. Invited talk.

[Jür03d] J. Jürjens. Developing safety-critical systems with UML. In Stevens [Ste03b], pages 360–372.

[Jür03e] J. Jürjens. Developing secure enterprise applications with UML. In *Fourth Workshop on UML for Enterprise Applications*. Object Management Group (OMG), 2003. Half-day tutorial.

[Jür03f] J. Jürjens. FoMSESS webpages. Working Group on Formal Methods and Software Engineering for Safety and Security (FoMSESS) of the German Computer Society (GI). Website at `http://www4.in.tum.de/~fomsess`, 2003.

[Jür03g] J. Jürjens. Model-based security with UMLsec. In *UML Forum Tokyo*, 2003. Invited talk.

[Jür04] J. Jürjens. UMLsec webpage, 2002-04. Accessible at `http://www.umlsec.org`. Protected content can be accessed as user: Reader, with password: Ihavethebook.

[Jür04a] J. Jürjens. Componentware for critical systems. *Journal of Object Technology*, 2004. To be published.

[Jür04b] J. Jürjens. Critical Systems Development with UML, 2004. Series of about 30 tutorials at international conferences 2002–04. Further information and download of material at `http://www4.in.tum.de/~juerjens/csdumltut`.

[Jür04c] J. Jürjens. Developing high-assurance secure systems with UML: An electronic purchase protocol. In *Eighth IEEE International Symposium on High Assurance Systems Engineering (HASE 2004)*, pages 231–240. IEEE Computer Society, New York, 2004.

[Jür04d] J. Jürjens. Developing security-critical applications with UMLsec – a short walk-through. *Novatica*, 168, March/April 2004.

[Jür04e] J. Jürjens. Developing Security-Critical Systems with UML, 2004. Series of tutorials at 30 international conferences including OMG DOCsec 2002, IFIP SEC 2002, ETAPS 2003, OMG Workshop On UML for Enterprise Applications 2003, Formal Methods Symposium 2003, ASE 2003, FMOODS 2003, ECOOP 2004, and WCC 2004. Download of material at `http://www4.in.tum.de/~juerjens/csdumltut`.

[Jür04f] J. Jürjens. Foundations for designing secure architectures. In *First International Workshop on Views On Designing Complex Architectures (VODCA 2004)*, Bertinoro, 2004.

[Jür04g] J. Jürjens. Logic for security. In *Spring School "Logic in Computer Science"*, Venezia, 2004. Graduate school on Logic in Informatics, Munich.

[Jür04h] J. Jürjens. Playing the devil's advocate: Testing real-time systems. In *Conference on Software Testing (ICSTEST-E 2004)*, Bilbao, 2004.

[Jür04i] J. Jürjens. Security in UML. In *4th International School on Foundations of Security Analysis and Design (FOSAD 2004)*, 2004.

[Jür04j] J. Jürjens. Standards and processes for modelbased engineering of safety- and security-critical systems. In *23rd International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2004)*, Lecture Notes in Computer Science. Springer, Berlin Heidelberg New York, 2004. Full-day tutorial.

[Jür04k] J. Jürjens. Tools for Critical Systems Development with UML. In *19th International Conference on Automated Software Engineering (ASE 2004)*. IEEE Computer Society, New York, 2004.

[Jür05] J. Jürjens. Security modeling with UML. Universidad Carlos III de Madrid, 2005.

[JW01a] J. Jürjens and G. Wimmel. Formally testing fail-safety of electronic purse protocols. In *16th International Conference on Automated Software Engineering (ASE 2001)*, pages 408–411. IEEE Computer Society, New York, 2001.

[JW01b] J. Jürjens and G. Wimmel. Security modelling for electronic commerce: The Common Electronic Purse Specifications. In B. Schmid, K. Stanoevska-Slabeva, and V. Tschammer, editors, *Towards the E-Society: E-Commerce, E-Business, and E-Government*, pages 489–506. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2001. First IFIP Conference on E-Commerce, E-Business, and E-Government (I3E 2001).

[JW02] J. Jürjens and G. Wimmel. Specification-based testing of firewalls. In D. Bjørner, M. Broy, and A. Zamulin, editors, *Andrei Ershov 4th International Conference "Perspectives of System Informatics" (PSI 2001)*, volume 2244 of *Lecture Notes in Computer Science*, pages 308–316. Springer, Berlin Heidelberg New York, 2002.

[KAH99] J. Kirby, M. Archer, and C. Heitmeyer. Applying formal methods to an information security device: An experience report. In *4th IEEE International Symposium on High Assurance Systems Engineering (HASE 1999)*, pages 81–88. IEEE Computer Society, New York, 1999.

[Kar00] G. Karjoth. Java and mobile code security – an operational semantics of Java 2 access control. In *Computer Security Foundations Workshop (CSFW)*, pages 224–232, 2000.

[Kem89] R. A. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, May 1989.

[KER99] S. Kent, A. Evans, and B. Rumpe. UML semantics FAQ. In A. Moreira and S. Demeyer, editors, *Object-Oriented Technology, ECOOP 1999 Workshop Reader*, volume 1743 of *Lecture Notes in Computer Science*, pages 33–56. Springer, Berlin Heidelberg New York, 1999.

[KG98] L. Kassab and S. Greenwald. Towards formalizing the Java Security Architecture in JDK 1.2. In Quisquater et al [QDMG98], pages 191–207.

[KK04] R. Kilian-Kehr. Can formal verification become mainstream in software engineering ? In J. Jürjens, editor, *FoMSESS 2004*, 2004. Second Workshop of the Working Group on Formal Methods and Software Engineering for Safety and Security (FoMSESS) of the German Computer Society (GI). Available at `http://www4.in.tum.de/~fomsess`.

[KN98] V. Kessler and H. Neumann. A sound logic for analysing electronic commerce protocols. In Quisquater et al [QDMG98], pages 345–360.

[Kna99] A. Knapp. A formal semantics for UML interactions. In R. B. France and B. Rumpe, editors, « *UML* »*'99: Second International Conference on The Unified Modeling Language*, volume 1723 of *Lecture Notes in Computer Science*, pages 116–130. Springer, Berlin Heidelberg New York, 1999.

[KPP02] M. Koch and F. Parisi-Presicce. Access control policy specification in UML. In Jürjens et al [JCF+02], pages 63–78.

[KRFL04] D.-K. Kim, I. Ray, R. B. France, and Na Li. Modeling role-based access control using parameterized UML models. In Wermelinger and Margaria [WM04], pages 180–193.

[Krü00] I. H. Krüger. *Distributed System Design with Message Sequence Charts*. PhD thesis, Institut für Informatik, TU München, 2000.

[Krü02] I. H. Krüger. Towards precise service specification with UML and UML-RT. In Jürjens et al [JCF+02], pages 19–34.

[KW01] A. Kleppe and J. Warmer. Unification of Static and Dynamic Semantics of UML. Technical report, Klasse Objecten, NL, 2001. Download at `http://www.klasse.nl/english/uml/uml-semantics.html`.

[KW04] R. Küsters and T. Wilke. Automata-based analysis of recursive cryptographic protocols. In V. Diekert and M. Habib, editors, *21st Annual Symposium on Theoretical Aspects of Computer Science (STACS 2004)*, volume 2996 of *Lecture Notes in Computer Science*, pages 382–393. Springer, Berlin Heidelberg New York, 2004.

[Lam73] B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[LB73] L. J. LaPadula and D. E. Bell. Secure computer systems: A mathematical model. Technical report, The MITRE Corporation, 1973. Reprinted in *Journal of Computer Security*, 4:239–263, 1996.

[LBD02] T. Lodderstedt, D. Basin, and J. Doser. SecureUML: A UML-based modeling language for model-driven security. In Jézéquel et al [JHC02], pages 426–441.

[LGS01] U. Lang, D. Gollmann, and R. Schreiner. Verifiable identifiers in middleware security. In *17th Annual Computer Security Applications Conference*, pages 450–459. IEEE Computer Society, New York, 2001.

[LHPP02] G. B. Lamont, H. Haddad, G. Papadopoulos, and B. Panda, editors. *Proceedings of the 2002 Symposium of Applied Computing (SAC)*. ACM Press, 2002.

[LLK+02] W. Längst, A. Lapp, K. Knorr, H.-P. Schneider, J. Schirmer, D. Kraft, and U. Kiencke. CARTRONIC-UML models: Basis for partially automated risk analysis in early development phases. In Jürjens et al [JCF+02], pages 3–17.

[LMM03] L. Lavazza, A. Morzenti, and S. Morasca. A dual language approach to the development of time-critical systems with UML. In Jürjens et al [JRFF03], pages 113–120.

[Lot97] V. Lotz. Threat scenarios as a means to formally develop secure systems. *Journal of Computer Security 5*, pages 31–67, 1997.

[Lot00] V. Lotz. Formally defining security properties with relations on streams. *Electronic Notes in Theoretical Computer Science*, 32, 2000. DERA/RHUL Workshop on Secure Architecture and Information Flow, 1999.

[Low95] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, November 1995.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. *Software Concepts and Tools*, 17(3):93–102, 1996.

[Low98] G. Lowe. Casper: A compiler for the analysis of security protocols. *Journal of Computer Security*, 6(1–2):53–84, 1998.

[LP99] J. Lilius and I. Porres. Formalising UML state machines for model checking. In R. B. France and B. Rumpe, editors, *The Unified Modeling Language (UML 1999)*, volume 1723 of *Lecture Notes in Computer Science*, pages 430–445. Springer, Berlin Heidelberg New York, 1999.

[LR97] G. Lowe and B. Roscoe. Using CSP to detect errors in the TMN protocol. *IEEE Transactions on Software Engineering*, 23(10):659–669, 1997.

[LW94] B. Liskov and J. Wing. A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems*, 16(6):1811–1841, November 1994.

[Man01] H. Mantel. Preserving information flow properties under refinement. In *Symposium on Security and Privacy (S&P)*, pages 78–93, 2001.

[Man02] H. Mantel. On the composition of secure systems. In *2002 IEEE Symposium on Security and Privacy*, pages 88–101. IEEE Computer Society, New York, 2002.

[MCF87] J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2):274–288, February 1987.

[McG98] G. McGraw. Testing for security during development: Why we should scrap penetrate-and-patch. *IEEE Aerospace and Electronic Systems*, April 1998.

[McL94] J. McLean. Security models. In J. Marciniak, editor, *Encyclopedia of Software Engineering*. John Wiley & Sons, New York, 1994.

[McL96] J. McLean. A general theory of composition for a class of "possibilistic" properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, 1996.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic, Boston, 1993.

[Mea91] C. Meadows. A system for the specification and analysis of key management protocols. In *Symposium on Security and Privacy (S&P)*, pages 182–195, 1991.

[Mea92] C. Meadows. Using traces based on procedure calls to reason about composability. In *Symposium on Security and Privacy (S&P)*, pages 177–188. IEEE Computer Society, New York, 1992.

[Mea95] C. Meadows. Formal verification of cryptographic protocols: A survey. In J. Pieprzyk and R. Safavi-Naini, editors, *Asiacrypt 1994*, volume

917 of *Lecture Notes in Computer Science*, pages 135–150. Springer, Berlin Heidelberg New York, 1995.

[Mea96] C. Meadows. The NRL Protocol Analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[Mea00] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, pages 237–250. IEEE Computer Society, New York, 2000.

[Mer02] S. Merz. From diagrams to semantics – and back (invited talk). In Jürjens et al [JCF+02].

[MGM03] H. Mouratidis, P. Giorgini, and G. A. Manson. Integrating security and systems engineering: Towards the modelling of secure information systems. In J. Eder and M. Missikoff, editors, *15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)*, volume 2681 of *Lecture Notes in Computer Science*, pages 63–78. Springer, Berlin Heidelberg New York, 2003.

[MH98] P. Malacaria and C. Hankin. Generalised flowcharts and games. *Lecture Notes in Computer Science*, 1443:363–??, 1998.

[MIB98] M. Maia, V. Iorio, and R. Bigonha. Interacting Abstract State Machines. In GI [GI98], pages 37–49.

[Mic01] Microsoft TechNet. Information about virus-infected hotfixes, April 25 2001. Available at `http://www.microsoft.com/technet/security/topics/virus/vihotfix.mspx`.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *2nd International Joint Conference on Artificial Intelligence (IJCAI 1971)*, pages 481–489, 1971.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.

[MIL+97] M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schumann, and K. Mayr. SETHEO and E-SETHEO – The CADE-13 Systems. *Journal of Automated Reasoning (JAR)*, 18(2):237–246, 1997.

[ML02] R. Marcano and N. Levy. Transformation rules of OCL constraints into B formal expressions. In Jürjens et al [JCF+02], pages 147–154.

[MMS97] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murφ. In *Symposium on Security and Privacy (S&P)*, pages 141–151. IEEE Computer Society, New York, 1997.

[MOF02] Object Management Group. *MOF 1.4 Specification*, April 2002. Available at `http://www.omg.org/technology/documents/formal/mof.htm`.

[MS00] É. Meyer and T. Santen. Behavioural conformance verification in an integrated approach using UML and B. In Grieskamp et al [GSS00], pages 358–379.

[MS03] H. Mantel and A. Sabelfeld. A unifying approach to the security of distributed and multi-threaded programs. *Journal of Computer Security*, 11(4):615–676, 2003.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1996. Download at `http://www.cacr.math.uwaterloo.ca/hac`.

[NEP01] NEPTUNE, 2001. `http://neptune.irit.fr`.

[Net03] Netbeans project. Open source. Available from `http://mdr.netbeans.org`, 2003.

[NNS02] F. Nielson, H. R. Nielson, and H. Seidl. Normalizable horn clauses, strongly recognizable relations, and spi. In M. V. Hermenegildo and G. Puebla, editors, *9th International Symposium on Static Analysis (SAS 2002)*, volume 2477 of *Lecture Notes in Computer Science*, pages 20–35. Springer, Berlin Heidelberg New York, 2002.

[NPW02] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*. Springer, Berlin Heidelberg New York, 2002.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[NSU03] Novosoft NSUML project. Available from `http://nsuml.sourceforge.net/`, 2003.

[Off95] J. Offutt. Practical mutation testing. In *12th International Conference on Testing Computer Software*, 1995.

[ÖP00] G. Övergaard and K. Palmkvist. Interacting subsystems in UML. In Evans et al [EKS00], pages 359–368.

[OXL99] J. Offutt, Y. Xiong, and S. Liu. Criteria for generating specification-based tests. In *5th IEEE Conference on Engineering of Complex Computer Systems (ICECCS 1999)*, page 119ff. IEEE Computer Society, New York, 1999.

[Pat02] A. Pataricza. From the general resource model to a general fault modeling paradigm? In Jürjens et al [JCF+02], pages 163–170.

[Pau98a] L. C. Paulson. Inductive analysis of the Internet protocol TLS (transcript of discussion). In B. Christianson, B. Crispo, W. S. Harbison, and M. Roe, editors, *Security Protocols*, volume 1550 of *Lecture Notes in Computer Science*, pages 13–23. Springer, Berlin Heidelberg New York, 1998. 6th International Workshop.

[Pau98b] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1–2):85–128, 1998.

[Pet02] L. Petre. Control systems development: A case study. In Jürjens et al [JCF+02], pages 171–180.

[PJWB03] G. Popp, J. Jürjens, G. Wimmel, and R. Breu. Security-critical system development with extended use cases. In *10th Asia-Pacific Software Engineering Conference (APSEC 2003)*, pages 478–487. IEEE Computer Society, New York, 2003.

[PMP01] Z. Pap, I. Majzik, and A. Pataricza. Checking general safety criteria on UML statecharts. In U. Voges, editor, *SAFECOMP 2001*, volume 2187 of *Lecture Notes in Computer Science*, pages 46–55. Springer, Berlin Heidelberg New York, 2001.

[Pol92] W. T. Polk. Automated tools for testing computer systems vulnerability. In *NIST Special Publications*. National Institute of Standards and Technology, December 1992.

[PR94] B. Paech and B. Rumpe. A new concept of refinement used for behaviour modelling with automata. In M. Naftalin, B. T. Denvir, and M. Bertran, editors, *FME 1994: Industrial Benefit of Formal Methods*, volume 873 of *Lecture Notes in Computer Science*, pages 154–174.

Springer, Berlin Heidelberg New York, 1994. Second International Symposium of Formal Methods Europe.

[PS97] J. Peleska and M. Siegel. Test automation of safety-critical reactive systems. *South African Computer Jounal*, 19:53–77, 1997.

[PW00] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *7th ACM Conference on Computer and Communications Security (CCS 2000)*, pages 245–254, 2000.

[QDMG98] J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors. *European Symposium on Research in Computer Security (ESORICS)*, volume 1485 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 1998.

[RACH00] G. Reggio, E. Astesiano, C. Choppy, and H. Hußmann. Analysing UML active classes and associated state machines – A lightweight formal approach. In T. S. E. Maibaum, editor, *Fundamental Approaches to Software Engineering (FASE 2000)*, volume 1783 of *Lecture Notes in Computer Science*, pages 127–146. Springer, Berlin Heidelberg New York, 2000.

[RCA00] G. Reggio, M. Cerioli, and E. Astesiano. An algebraic semantics of UML supporting its multiview approach. In D. Heylen, A. Nijholt, and G. Scollo, editors, *Algebraic Methods in Language Processing (AMiLP 2000)*, number 16 in International Twente Workshop on Language Technology (TWLT). University of Twente, NL, 2000. Algebraic Methodology and Software Technology (AMAST) Workshop.

[RCA01] G. Reggio, M. Cerioli, and E. Astesiano. Towards a rigorous semantics of UML supporting its multiview approach. In Hußmann [Huß01], pages 171–186.

[RE00] W. Rankl and W. Effing. *Smart Card Handbook*. John Wiley & Sons, New York, 2000. 2nd edition.

[RFBLO01] D. Riehle, S. Fraleigh, D. Bucka-Lassen, and N. Omorogbe. The architecture of a UML virtual machine. In *ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA 2001)*, volume 36 (11) of *SIGPLAN Notices*, pages 327–341. ACM, New York, November 2001.

[RFLG03] I. Ray, R. B. France, N. Li, and G. Georg. An aspect-based approach to modeling access control concerns. *Information & Software Technology*, 2003. To be published.

[Ric03] R. Richardson. 2003 CSI/FBI computer crime and security survey. Technical report, Computer Security Institute, San Francisco, May 2003. Available at `http://www.gocsi.com/forms/fbi/pdf.html`.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Reading, MA, 1999.

[RJW+03] J. Romberg, J. Jürjens, G. Wimmel, O. Slotosch, and G. Hahn. AutoFocus and the MoDe Tool. In *3rd International Conference on Application of Concurrency to System Design (ACSD 2003)*, pages 249–250. IEEE Computer Society, New York, 2003.

[RLKF03] I. Ray, N. Li, D.-K. Kim, and R. B. France. Using parameterized UML to specify and compose access control models. In *6th Working Conference on Integrity and Internal Control in Information Systems (IICIS)*. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2003.

[RS98] P. Ryan and S. Schneider. An attack on a recursive authentication protocol. *Information Processing Letters*, 65:7–10, 1998.

[RS01] A. Rosenthal and E. Sciore. Administering permissions for distributed data: Factoring and automated inference. In *Conference on Data and Application Security*, pages 91–104. International Federation for Information Processing (IFIP), 2001.

[RSG+01] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The Modelling and Analysis of Security Protocols: the CSP Approach.* Addison-Wesley, Reading, MA, 2001.

[Rum96] Bernhard Rumpe. *Formale Methodik des Entwurfs verteilter objektorientierter Systeme.* PhD thesis, TU München, 1996.

[Rum02] B. Rumpe. Executable modeling with UML. A vision or a nightmare? In *Issues & Trends of Information Technology Management in Contemporary Associations*, pages 697–701, Seattle, 2002. Idea Group Publishing, London.

[Rum04] B. Rumpe. *Modellierung mit UML.* Springer, Berlin Heidelberg New York, 2004.

[RV01] A. Riazanov and A. Voronkov. Vampire 1.1 (system description). In R. Goré, A. Leitsch, and T. Nipkow, editors, *First International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Computer Science*, pages 376–380. Springer, Berlin Heidelberg New York, 2001.

[RW99] G. Reggio and R. Wieringa. Thirty one problems in the semantics of UML 1.3 dynamics. In *Rigorous Modelling and Analysis of the UML: Challenges and Limitations*, 1999. OOPSLA 1999 workshop.

[RWW94] A. Roscoe, J. Woodcock, and L. Wulf. Non-interference through determinism. In D. Gollmann, editor, *3rd European Symposium on Research in Computer Security (ESORICS 1994)*, volume 875 of *Lecture Notes in Computer Science*, pages 33–53. Springer, Berlin Heidelberg New York, 1994.

[SBCJ97] P. Samarati, E. Bertino, A. Ciampichetti, and S. Jajodia. Information flow control in object-oriented systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 524–537, August 1997.

[SC01] P. F. Syverson and I. Cervesato. The logic of authentication protocols. In *Foundations of Security Analysis and Design (FOSAD 2000)*, volume 2171 of *Lecture Notes in Computer Science*, pages 63–136, 2001.

[Sch96] S. Schneider. Security properties and CSP. In *IEEE Symposium on Security and Privacy (S&P)*, pages 174–187, 1996.

[Sch97] J. Schumann. Automatic verification of cryptographic protocols with SETHEO. In W. McCune, editor, *14th International Conference on Automated Deduction (CADE-14)*, volume 1249 of *Lecture Notes in Computer Science*, pages 87–100. Springer, Berlin Heidelberg New York, 1997.

[Sch98] W. Schönfeld. Interacting Abstract State Machines. In GI [GI98], pages 22–36.

[Sch99a] F. Schneider, editor. *Trust in Cyberspace.* National Academy Press, Washington, DC, 1999. Available at `http://www.nap.edu/readingroom/books/trust`.

[Sch99b] J. Schumann. PIL/SETHEO: A tool for the automatic analysis of authentication protocols. In N. Halbwachs and D. Peled, editors, *11th*

International Conference on Computer Aided Verification (CAV '99), volume 1633 of *Lecture Notes in Computer Science*, pages 500–504. Springer, Berlin Heidelberg New York, 1999.

[Sch00] W. Schulte. Why doesn't anyone use formal methods? In Grieskamp et al [GSS00], pages 297–298.

[Sch01] J. M. Schumann. *Automated Theorem Proving in Software Engineering*. Springer, Berlin Heidelberg New York, 2001.

[Sch03a] M. Schumacher. *Security Engineering with Patterns – Origins, Theoretical Models, and New Applications*, volume 2754 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2003.

[Sch03b] M. A. Schwaiger. Tool-supported analysis of business processes and SAP permissions, 2003. Study project, TU München.

[SCW00] S. Stepney, D. Cooper, and J. Woodcock. *An Electronic Purse: Specification, Refinement, and Proof*. Oxford University Computing Laboratory, 2000. Technical Monograph PRG-126.

[Sel03] B. Selic. The use of modeling techniques in safety-critical system design (invited talk). In Jürjens et al [JRFF03], pages 1–2.

[SG03] E. Snekkenes and D. Gollmann, editors. *8th European Symposium on Research in Computer Security (ESORICS 2003)*, volume 2808 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2003.

[SH99] B. Schätz and F. Huber. Integrating formal description techniques. In J. M. Wing, J. Woodcock, and J. Davies, editors, *World Congress on Formal Methods in the Development of Computing Systems (FM 1999)*, volume 1709 of *Lecture Notes in Computer Science*, pages 1206–1225. Springer, Berlin Heidelberg New York, 1999.

[Sha99] A. Shamir. Crypto predictions. In *3rd International Conference on Financial Cryptography (FC 1999)*, 1999.

[SHP02] T. Santen, M. Heisel, and A. Pfitzmann. Confidentiality-preserving refinement is compositional – sometimes. In Gollmann et al [GKW02], pages 194–211.

[SKM01] T. Schäfer, A. Knapp, and S. Merz. Model checking UML state machines and collaborations. In S. D. Stoller and W. Visser, editors, *Workshop on Software Model Checking*, volume 55(3) of *Electronic Notes in Theoretical Computer Science*. Elsevier, Amsterdam, 2001. Satellite event of the 13th International Conference on Computer-Aided Verification (CAV 2001)

[SP99] P. Stevens and R. Pooley. *Using UML: software engineering with objects and components*. Addison-Wesley Longman, Reading, MA, 1999.

[SR98] B. Selic and J. Rumbaugh. Using UML for modeling complex real-time systems. Available at `http://www-106.ibm.com/developerworks/rational/library/`, 1998.

[SS75] J. Saltzer and M. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, September 1975.

[SS94] R. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[SS01] G. Sutcliffe and C. Suttner. The tptp problem library for automated theorem proving, 2001. Available at `http:\\www.tptp.org`.

[SSB01] R. Stärk, J. Schmid, and E. Börger. *Java and the Java virtual machine – definition, verification, validation*. Springer, Berlin Heidelberg New York, 2001.

[SSRB00] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-oriented Software Architecture*, volume 2. John Wiley & Sons, New York, 2000. Patterns for Concurrent and Networked Objects.

[ST00] S. F. Smith and C. L. Talcott, editors. *4th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS 2000)*, volume 177 of *IFIP Conference Proceedings*. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2000.

[Ste01a] P. Stevens. On use cases and their relationships in the Unified Modelling Language. In Hußmann [Huß01], pages 140–155.

[Ste01b] P. Stevens. A revolution in UML tool use? Tool adaptation, extension and integration using XMI. UML 2001 tutorial, 2001.

[Ste03a] P. Stevens. Small-scale XMI programming; a revolution in UML tool use? *Journal of Automated Software Engineering*, 10(1):7–21, 2003. Kluwer.

[Ste03b] P. Stevens, editor. *The Unified Modeling Language (UML 2003)*, volume 2863 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2003. 6th International Conference.

[Stø01] K. Stølen. A framework for risk analysis of security critical systems. In *Supplement of the 2001 International Conference on Dependable Systems and Networks (DSN 2001)*, pages D4–D11, 2001.

[Stö03] H. Störrle. Assert and negate and refinement in UML-2 interactions. In Jürjens et al [JRFF03], pages 79–94.

[SvO94] P. Syverson and P. van Oorschot. On unifying some cryptographic protocol logics. In *IEEE Symposium on Security and Privacy*, pages 14–28, 1994.

[SW97] A. Schürr and A. Winter. Formal definition and refinement of UML's module/package concept. In J. Bosch and S. Mitchell, editors, *Object-Oriented Technology (ECOOP'97 Workshop Reader)*, volume 1357 of *Lecture Notes in Computer Science*, pages 211–215. Springer, Berlin Heidelberg New York, 1997.

[SW00] G. Stenz and A. Wolf. E-SETHEO: An automated^3 theorem prover. In R. Dyckhoff, editor, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2000)*, volume 1847 of *Lecture Notes in Computer Science*, pages 436–440. Springer, Berlin Heidelberg New York, 2000.

[TFHG99] F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

[The01] The Register. Microsoft security fixes infected with funlove virus, April 25 2001. Available at `http://www.theregister.co.uk/2001/04/25/microsoft_security_fixes_infected`.

[TS02] J. Tenzer and P. Stevens. Modelling recursive calls with UML state diagrams. In M. Pezzè, editor, *Fundamental Approaches to Software Engineering (FASE 2003)*, volume 2621 of *Lecture Notes in Computer Science*, pages 135–149. Springer, Berlin Heidelberg New York, 2002.

[UML03] Object Management Group. *OMG Unified Modeling Language Specification v1.5*, March 2003. Version 1.5. OMG Document formal/03-03-01.

[vdB02] M. von der Beeck. A structured operational semantics for UML-statecharts. *Software and System Modeling*, 1(2):130–141, 2002.

[Ver03] Verisoft project webpages. `http://www.verisoft.de`, 2003.

[vKSZ03] J. van Katwijk, B. Sanden, and J. Zalewski. A study of new approaches to evaluate real-time software architectures for safety-critical systems. In Jürjens et al [JRFF03], pages 121–128.

[VM98] J. Voas and G. McGraw. *Software Fault Injection: Inoculating Programs Against Errors*. John Wiley & Sons, New York, 1998.

[VM02] J. Viega and G. McGraw. *Building Secure Software*. Addison-Wesley, Reading, MA, 2002.

[vOL02] D. von Oheimb and V. Lotz. Formal security analysis with interacting state machines. In Gollmann et al [GKW02], pages 212–228.

[VP03] D. Varro and A. Pataricza. Automated formal verification of model transformations. In Jürjens et al [JRFF03], pages 63–78.

[VWW02] M. Vetterling, G. Wimmel, and A. Wisspeintner. Secure systems development based on the common criteria. In *10th International Symposium on the Foundations of Software Engineering (FSE-10)*, pages 129–138. ACM, New York, 2002.

[Wal00] M. Walker. On the security of 3GPP networks. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 102–103. Springer, Berlin Heidelberg New York, 2000.

[Wat02] B. Watson. Non-functional analysis for UML models. In *Real-Time and Embedded Distributed Object Computing Workshop*. OMG, July 15–18, 2002. `http://www.omg.org/news/meetings/workshops/RT_2002_Workshop_Presentations/03-1_Watson_Non-FunctionalAnalysis.pdf`.

[WBH+02] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobald, and D. Topić. Spass version 2.0. In *18th International Conference on Automated Deduction (CADE-18)*, volume 2392 of *Lecture Notes in Computer Science*, pages 275–279. Springer, Berlin Heidelberg New York, 2002.

[Weh02] H. Wehrheim. Checking behavioural subtypes via refinement. In B. Jacobs and A. Rensink, editors, *5th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS 2002)*, pages 79–94. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2002.

[Wei95] C. Weissman. Penetration testing. In Abrams et al [AJP95], chapter 11, pages 269–296.

[Wei99] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In H. Ganzinger, editor, *16th International Conference on Automated Deduction (CADE-16)*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328, 1999.

[WF98] D. S. Wallach and E. W. Felten. Understanding Java stack inspection. In *IEEE Symposium on Security and Privacy (S&P)*, page 52ff, 1998.

[Whi00] J. Whittle. Formal approaches to systems analysis using UML: An overview. *Journal of Database Management*, 11(4):4–13, 2000.

[Wir71] N. Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14(4):221–227, April 1971. Available at `http://www.acm.org/classics/dec95`.

[WJ02] G. Wimmel and J. Jürjens. Specification-based test generation for security-critical systems using mutations. In *International Conference on Formal Engineering Methods (ICFEM)*, volume 2495 of *Lecture Notes in Computer Science*, pages 471–482. Springer, Berlin Heidelberg New York, 2002.

[WM04] M. Wermelinger and T. Margaria, editors. *Fundamental Approaches to Software Engineering (FASE 2000)*, volume 2984 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg New York, 2004.

[WS02] A. Wasowski and P. Sestoft. Compile-time scope resolution for statecharts transition. In Jürjens et al [JCF+02], pages 133–146.

[WW01] G. Wimmel and A. Wißpeintner. Extended description techniques for security engineering. In M. Dupuy and P. Paradinas, editors, *Trusted Information: The New Decade Challenge*, page 469ff. International Federation for Information Processing (IFIP), Kluwer Academic, Dordrecht, 2001. 16th International Conference on Information Security (SEC 2001).

[XLL03] Zhongfu Xu, J. Luethi, and A. Lehmann. Predicting software performance based on UML models during the unified software development process. In Jürjens et al [JRFF03], pages 105–112.

[XMI02] Object Management Group. *OMG XML Metadata Interchange (XMI) Specification*, January 2002.

[ZG02] P. Ziemann and M. Gogolla. An extension of OCL with temporal logic. In Jürjens et al [JCF+02], pages 53–62.

Index