Security Modelling for Electronic Commerce: The Common Electronic Purse Specifications*

Jan Jürjens[1] and Guido Wimmel[2]
[1]*Computing Laboratory, University of Oxford, email: jan@comlab.ox.ac.uk*
[2]*Dept. of Computer Science, Munich University of Technology, email: wimmel@in.tum.de*

Abstract: Designing security-critical systems correctly is very difficult. We present work on software engineering of security critical systems, supported by the CASE tool AUTOFOCUS.

Security critical systems are specified with extended structure diagrams, message sequence charts for the protocols and statecharts for the attacker, translated into an AUTOFOCUS system model and examined for security weaknesses using model checking. Additionally, the specifications could be simulated or tested - which is a first step towards integration of cryptographic primitives, intuitive graphical modelling, simulation and model checking.

We explain our method at the example of a part of the Common Electronic Purse Specifications (CEPS), and comment on potential of vulnerability and consequences for the design.

1. INTRODUCTION

Security aspects have become an increasingly important issue in developing distributed systems, especially in the electronic business sector. Because failures of security mechanisms may cause very high potential damage (e.g., loss of money through fraud), the correctness of such systems is crucial.

* This work was partially supported by the Studienstiftung des deutschen Volkes, and by the German Ministry of Economics within the FairPay project

Designing security critical systems correctly is difficult. Also, it is easy to misunderstand assumptions on the environment in which e.g. protocols are to be used and what their secure functioning may rely on. Security violations often occur at the boundaries between security mechanisms and the general system (Gollmann, 1999; Anderson, 2001).

Therefore, the consideration of security aspects has to be integrated into general systems development (Anderson, 1994). Common modelling techniques used in industry, such as collaboration diagrams, state charts and message sequence charts have to be tailored for that purpose. For instance, (Jürjens, 2001c) presents first work towards that goal by extending the UML (Unified Modelling Language).

On the other hand, to ensure the correctness of the systems, the models have to be sufficiently precise, to be able to state security properties in an unambiguous way and to formally verify their truth or find possible weaknesses, using mathematical reasoning or automated verification with model checkers.

In this work, we show how to model and reason about a security-critical protocol (the purchase transaction) of the Common Electronic Purse Specifications CEPS (CEPSCO, 2000), supported by the CASE tool AUTOFOCUS (Huber et al., 1998a; Huber et al., 1998b). CEPS is a candidate for a globally interoperable electronic purse standard supported by organisations (including Visa International) representing 90 percent of the world's electronic purse cards and likely to become an accepted standard (Asokan et al., 2000), making its security an important goal.

We specify cryptographic protocols using message sequence charts (MSCs), in a way similar to the usual informal notation of security protocols. These specifications can be translated mechanically into an AUTOFOCUS system model consisting of state transition diagrams (STDs) (Krüger, 2000). Together with the modelled adversary, this system is checked for security weaknesses automatically using the model checker SMV connected to AUTOFOCUS to verify the desired security properties of the protocol.

Our approach has the benefits of combining intuitive graphical modelling, simulation and model checking, and allows to represent counterexamples as MSCs. Since the AUTOFOCUS tool is closely related to the formal development method Focus (Broy et al., 1992), our approach also supports formal proofs in this framework, allowing to build on results such as (Lotz, 1997). The intruder model used is rather flexible, the adversary can switch between acting as one or another party, intercept only certain messages or learn certain keys etc.

In the following subsection we present some background information and refer to related work. In Section 2, we give an overview over the Common Electronic Purse Specifications, specify the part under

consideration and explain the security threat model. In Section 3, we introduce the notation of AUTOFOCUS and use it to investigate the above specification. We end with a conclusion and indicate further planned work.

1.1 Security-assurance using formal tools

There has been extensive research in using formal methods to verify security protocols, following an abstract way to describe protocols in (Dolev and Yao, 1983). A few examples are (Burrows et al., 1989; Lowe 1996; Paulson 1998; Pfitzmann and Waidner 2001); an overview is in (Gritzalis et al., 1999; Ryan et al., 2001). Smart card protocols have been investigated using formal logic in (Abadi et al., 1993). (Gollmann, 2000) points out the need to consider the underlying physical layer in formal security investigations.

While many case-studies consider protocols from the academic literature (usually presented in a much more tractable form), notable examples of verifications of smart-card payment system used in practice can be found in (Anderson, 1999; Stepney et al., 2000).

As an example for the treatment of security in the context of general systems engineering, (Jürjens, 2001c) presents work towards using the UML notation in security engineering which is applied to the CEPS load transaction in (Jürjens, 2001a).

AUTOFOCUS has been used for security in (Wimmel and Wißpeintner 2001), the underlying Focus model in (Lotz, 1997).

2. CEPS

We give an overview over the Common Electronic Purse Specifications and specify the (simplified) purchase transaction to be investigated.

Stored value smart cards (“electronic purses”) have been proposed to allow cash-free point-of-sale (POS) transactions offering more fraud protection than credit cards: Their built-in chip can perform cryptographic operations which allows transaction-bound authentication[1] (whereas credit card numbers are valid until the card is stopped, enabling misuse). The card contains an account balance that is adjusted when loading the card or purchasing goods.

Here we consider the central part of CEPS, the purchase transaction, which allows the cardholder to use the electronic value on a card to pay for goods. The participants involved in the transaction protocol are the

[1]For a discussion of authentication cf. (Gollmann, 1996).

customer's card and the merchant's POS device. The POS device contains a Purchase Security Application Module (PSAM) (a smart-card) that is used to store and process data (and assumed to be tamper-resistant). During the transaction, the account balance in the card is decremented, and the balance in the PSAM is incremented by the corresponding amount. The card issuer later receives transaction logs.

In addition to transactions using public terminals it is intended to use CEPS-cards for transactions over the Internet (CEPSCO, 2000, Bus. Req.).

2.1 Specification of CEPS Purchase Transaction

Apart from incremental transactions not considered here, security functionality is provided only by the PSAM (and not the rest of the POS device). Thus our protocol participants are the CEP card C and the PSAM M (with public and private keys K_C and K_C^{-1} resp. K_M and K_M^{-1}, where the public keys are exchanged before the transaction). The protocol consists of the following steps (see Fig. 3 for a formal specification by AUTOFOCUS MSCs):

1. PSAM $\rightarrow$ Card: $\mathrm{Init}(\{\{SK_{PS}\}_{K_M^{-1}}\}_{K_C})$
2. Card $\rightarrow$ PSAM: $\mathrm{Resp}(\{S\}_{SK_{PS}})$
3. PSAM $\rightarrow$ Card: `OK`

The PSAM initiates the transaction after the CEP card is inserted into the POS device, by sending a message containing a freshly created session key SK_{PS} signed by M and encrypted with C's public key. Whenever the card receives a message after being inserted into the POS device, it tries to decrypt it with its private key and checks its signature with M's public key. If this is successful, C responds with a message containing S encrypted under the received session key (otherwise it waits for the next message). S contains identification data to authenticate the card and transaction data for logging purposes (and validation by the card issuer later). We assume that only C can produce S and that M can verify if a received message is such an S produced by C (in practice, this is achieved by having C sign the message) and therefore model S simply as a secret value (to simplify mechanical verification). Finally, after M receives a message encrypted under SK_{PS} it stores the contents and ends with a message `OK`. We leave out some message parts that are only relevant for logging.

2.2 Security Threat Model

The CEP specifications require the smart card and the PSAM (but not the POS device (CEPSCO, 2000, Bus Req. p. 13, Funct. req. p. 20) to be tamper-proof. The purchase transaction is supposed to provide mutual

authentication between the terminal and the card using a chain of certificates of which the first is issued by a Certification Authority and the last contains the card's or PSAM's public key.

The smart card is inserted into a POS device and can thus communicate with the PSAM. Since there is no direct communication between the cardholder and her card, the information displayed by the POS device regarding the transaction has to be trusted at the point of transaction. Security for the customer against fraud by the merchant is supposed to be provided through logging the (signed) transaction details and a posteriori settlement involving the card issuer. Similarly, security for the merchant against the customer is supposed to be provided by exchanging the purchased good only for a signed message from the card containing the transaction details, for which the merchant will receive the corresponding monetary amount from the issuer afterwards.

The idea is that risk of fraud is kept small since fraud should be detected in the settlement later and certificates of cards or PSAMs actively involved in fraud can be revoked.

In our formal investigation, we will consider the following two threat scenarios in order to see if they allow an attack: A sufficiently motivated adversary makes a POS device publicly available (in the case of ATMs, such cases are reported in (Anderson, 1994)) which only communicates with the card (to receive transaction information) and then returns the card with an error message, without actually having completed a transaction.

In the first scenario, the attacker then uses a smart card including the information obtained from the earlier interaction and tries to attack a merchant's POS device by buying goods with transaction messages signed by the earlier attacked card. If the attack succeeds, the attacker terminal or card do not show up in the audit trail at all, so the attacker cannot be made responsible.

This scenario corresponds to an attempted attack (also called "man-in-the-middle" attack) where the attacker first communicates with the attacked card (in the role of a PSAM) and then with the attacked PSAM (in the role of a card).

In the second, more sophisticated scenario the attacker could try to attack an inserted card as above, and in parallel a PSAM in a POS device set up by a merchant in an unsupervised place by tampering with the POS device (not assumed to be tamper-proof) in order to directly communicate with the PSAM. Via a radio connection the attacker could thus communicate both with the attacked card and the attacked PSAM (if he succeeds in synchronising the two events). Thus he could buy goods with transaction messages signed by the card attacked in parallel. Again the attacker terminal or card do not show up in the audit trail.

This scenario corresponds to an attack where the attacker communicates with the attacked card and the attacked PSAM in parallel.

This scenario is more realistic when using CEPS for transactions over the Internet, as intended (CEPSCO, 2000, Bus. Req.). Then, the mentioned synchronisation is much easier to realise, since the attacker needs to initiate his purchase over the Internet only when the attacked card is inserted into the modified terminal and can pay directly with the attacked card. Of course, the purchase should be anonymous (purchase of access to multimedia content).

3. INVESTIGATING CEPS WITH AUTOFOCUS

In this chapter, we show how to specify security critical systems using the CASE tool AUTOFOCUS/Quest (Huber et al., 1998a; Slotosch, 1998; Phillips and Slotosch, 1999) recently developed at Munich University of Technology with the goal to combine user-friendly graphical system design using common description techniques such as collaboration diagrams, state charts and message sequence charts, support of simulation, code generation and formal verification of correctness.

(Jürjens, 2001b) gives a specification language to represent cryptographic primitives in Focus (Broy et al., 1992), the formal foundation of AUTOFOCUS. On the basis of this and an earlier case study — an AUTOFOCUS model of the Needham-Schroeder public key protocol — we present a framework for AUTOFOCUS specifications of such systems. The Focus specification language in (Jürjens, 2001b) and the AUTOFOCUS model are closely related and can easily be translated into each other, so simulation, verification, code generation and model checking in AUTOFOCUS are supported as well as formal proofs using the Focus method.

System specifications in AUTOFOCUS make use of the following views:

- **System Structure Diagrams (SSDs)** are similar to data flow resp. collaboration diagrams and describe the structure and the interface of a system. In the SSD view, the system consists of a number of communicating components, which have input and output ports to allow for sending and receiving messages of a particular data type. The ports can be connected via channels, making it possible for the components to exchange data. SSDs can be hierarchical, i.e. a component belonging to an SSD can have a substructure that is defined by an SSD itself. Besides, the components in an SSD can be associated with local variables.
- **Data Type Definitions (DTDs)** define the data types used in the model, with the functional language Quest (Phillips and Slotosch, 1999). In addition to basic types as integer, user-defined hierarchic data types are offered that are similar to those in functional programming languages.

- **State Transition Diagrams (STDs)** represent extended finite automata and are used to describe the behaviour of a component in an SSD. The automata consist of a set States of states, and a set $Tr \subseteq States \times States \times PRE_EXP \times INP_EXP \times OUT_EXP \times POST_EXP$ of transitions, where
  - PRE_EXP are boolean terms on local variables and variables bound in INP_EXP representing a **precondition** for transition firing.
  - INP_EXP denotes **input patterns** of the form $inp_1?x;inp_2?y;...$ (i.e., reading values from input channels). The x,y can also be pattern matching expressions (which will be explained later).
  - OUT_EXP denotes **output expressions** of the form $out_1!term_1;out_2!term_2;...$ (output values of expressions to ports)
  - POST_EXP are **postconditions** of the form $lvar_1:=term_1;lvar_2:=term_2;...$ (sets local variables to the values of $term_i$, which can include local variables and variables bound in INP_EXP).
- In AUTOFOCUS, a transition (s,t,p,i,o,q) from s to t is annotated with $p:i:o:q$. Leaving out components is interpreted as **true** for pre-conditions, and as the empty list in the other cases. A transition is executable if the input patterns match the values at the input channels and the precondition is true. At each clock tick, one executable transition in each component fires, outputs the values specified by the output patterns and sets the local variables according to the postcondition. The values at the output ports can be read by the connected components in the next clock cycle.
- **Extended Event Traces (EETs)** finally make it possible to describe exemplary system runs, similar to MSCs (ITU, 1996).
- The Quest extensions (Slotosch, 1998) to AUTOFOCUS offer various connections to programming languages and formal verification tools, such as Java code generation, model checking using SMV, and bounded model checking and test case generation (Wimmel et al., 2000).

3.1 Specification of Security Critical Systems in AUTOFOCUS

In this section, we explain how to use AUTOFOCUS to specify security critical systems so that they can be examined for security weaknesses by presenting a model for the CEPS purchase transaction.

3.1.1 Abstract System Model

Figure 1 shows an abstract system model (high-level design) of the CEPS purchase transaction. The system consists of two components `Card`

and PSAM, which are connected via channels. In addition to conventional system structure diagrams, one can use security tags (Wimmel and Wißpeintner, 2001) to specify properties relevant for security evaluations. In this case, the channels are labelled with “public”, which means that they can be accessed by an intruder. Moreover, the protocol between the two parties is specified.

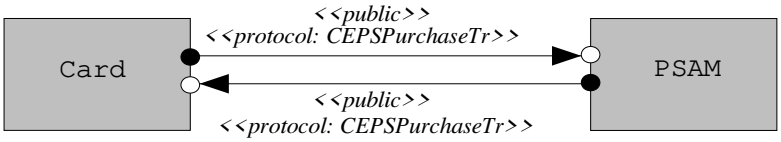

Figure 1. Abstract System Model for CEPS

The abstract system model must then be refined into system structure, behaviour and data type views to allow for concrete security analyses.

3.1.2 System Structure Diagram

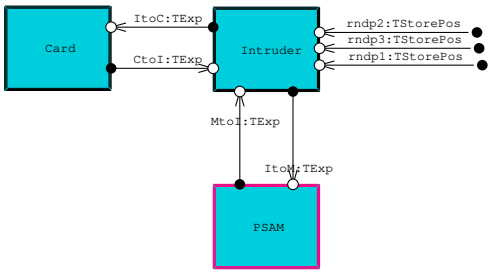

Figure 2. System Structure Diagram for CEPS System

Figure 2 shows the system structure diagram of the system. It corresponds to the system model shown in Fig. 2. However, there is now an additional component Intruder. As the channel between PSAM and Card is marked “public”, all messages between PSAM and Card has to pass this component, which thus can access and manipulate the messages. The additional channels rndi into the intruder are explained later.

3.1.3 Data Type Definition

The messages that can be sent through the channels are cryptographic expressions. A cryptographic expressions can be a basic element such as an

empty message, a key or a name, or an encrypted expression (under a certain key), or a concatenation of two expressions. This is represented by the following AUTOFOCUS data type definition:

```
data TKey = KM_ | KM | KC_ | KC | SKPS | SKI | S;
data TExp = Empty | Key(TKey) | Encr(TKey,TExp)
 | Concat(TExp,TExp);
```

The keys given in the definition of `TKey` correspond to the keys used in the specification shown in Section 2.1. `KC_` denotes K_C^{-1}, and an additional secret key `SKI` was added for the intruder.

However, to be able to use model-checking to examine vulnerabilities, the data types need to be finite. Therefore the recursive data type `TExp` must be replaced by a non-recursive one. This is straightforward to accomplish if we note that in our system we only have two possible types of valid messages: a message of the form $\{x\}_{K1}$ and a message $\{\{x\}_{K1}\}_{K2}$. We thus represent these by the new data types `TEncr1` and `TEncr2`. `TExp` can now either be `Empty` or consist of one of these data types:

```
data TExp = Empty | Exp1(exp1:TEncr1) | Exp2(exp2:TEncr2);
data TEncr1 = Encr1(keyenc1:TKey, expenc1:TKey);
data TEncr2 = Encr2(keyenc2:TKey, expenc2:TEncr1);
```

Expressions can now be represented by constructor terms. For example, `Exp1(Encr1(SKPS,S))` corresponds to $\{S\}_{SKPS}$, and `Exp2(Encr2(KM,Encr1(KC_,SKPS)))` corresponds to $\{\{SKPS\}_{KC}^{-1}\}_{KM}$. From the first message (say it is stored in a variable `x`), the key can be extracted using selectors: `keyenc1(exp1(x))` gives `SKPS`.

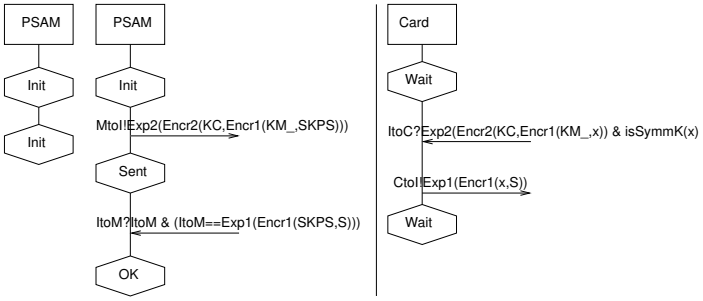

Figure 3. AutoFocus MSCs for CEPS System

3.1.4 Behaviour of the PSAM

The PSAM and the card execute the protocol *CEPSPurchaseTr*, which is specified using message sequence charts (MSCs). Figure 3(a),(b) show the

AUTOFOCUS MSCs specifying the behaviour of the PSAM (conf. Fig. 1). In this Figure, diamond shaped elements denote states. Thus, the PSAM can either wait in the `Init` state, or execute the protocol. Figure 4 shows the state transition diagram generated from this representation. MSCs are particularly suitable for specifying cryptograpic protocols as they represent sequential executions and correspond to the usual informal notation.

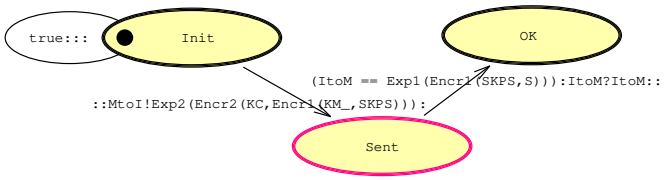

Figure 4. State Transition Diagram for the PSAM

3.1.5 Behaviour of the Card

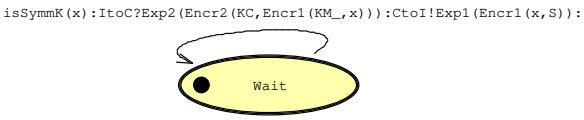

Figure 5. State Transition Diagram for the Card

Figure 4 shows the state transition diagram for the Card, generated from the specification in Fig. 3(c). This also demonstrates how *input patterns* can be used in AUTOFOCUS to make transition annotations more readable. The pattern `ItoC?Exp2(Encr2(KC,Encr1(KM_,x)))` is an abbreviation for `ItoC?ItoC` and the precondition

`is_Exp2(ItoC)` ∧ `is_Encr2(exp2(ItoC)` ∧ `keyenc2(exp2(ItoC))==KC`

and binding of the variable *x* to `expenc1(expenc2(exp2(ItoC)))`.

In addition, the card makes sure that the key sent to it is a symmetric one. This is done by checking *`isSymmK(x)`*, which is given as a *function definition*, the meaning of which should be obvious:

```
fun isSymmK(SKPS) = True
  | isSymmK(SKI) = True
  | isSymmK(x) = False;
```

3.1.6 The intruder model

The above model specifies the data types, system structure and behaviour of the involved parties for the CEPS purchase transaction. This system could now be simulated and tested using AUTOFOCUS.

To be able to investigate vulnerabilities, we use an intruder model commonly employed in formal reasoning about security protocols. All messages in the system are sent via the intruder, who can thus intercept them, learn secrets in the messages, and generate own messages or replay messages.

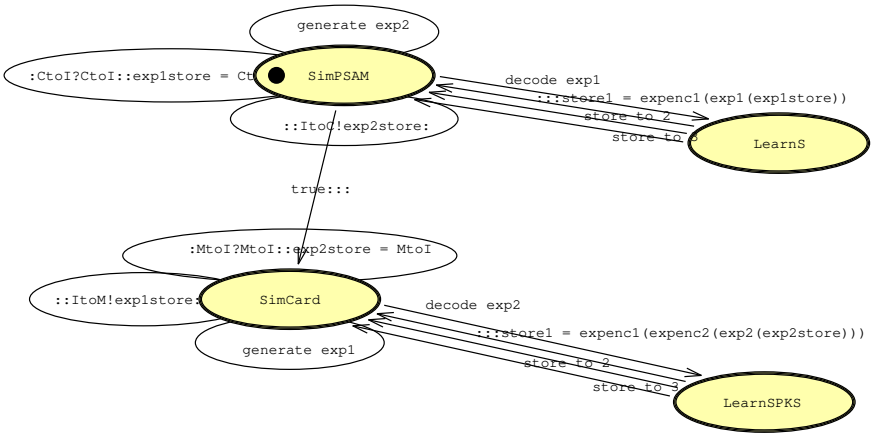

Figure 6. State Transition Diagram for the Intruder

As the intruder model is highly nondeterministic and allows many different executions, it is best specified by an STD (see Figure 6). For readability, some transitions are annotated with a text label instead of the full annotation (which is shown by the CASE tool on a mouse click).

The intruder can store two messages (local variables `exp1store` and `exp2store`), one of each type (`Exp1` and `Exp2`), and three keys (local variables `store1,2,3`). It has four states: `SimPSAM`, `SimCard`, `LearnS`, and `LearnSKPS`. In state `SimPSAM` it acts as the PSAM, reading messages from channel `CtoI` and storing them. The transition to `LearnS` labelled `decode exp1` can be executed if the intruder can decrypt the message in the store (which is of the form `Exp1(Encr(k,x))`) and learn the secret `x` contained in it. This is the case if the intruder has the key `k` somewhere in his key store. To model the searching of the store, the input channels `rndi` nondeterministically provide the intruder with a store position he can choose

a key from, to compare it with the key used for encryption. Altogether, the `decode exp1` transition is given as

```
(is_Exp1(exp1store) && (keyenc1(exp1(exp1store)) ==
retrkey(store1,store2,store3,p1))):rndp1?p1::
```

where `retrkey` is a function choosing one of `store1`, `store2` ,`store3` depending on p1. In the transitions back to `SimPSAM`, the content of the message is stored in one of the three possible places.

In a similar way, the transition `generate exp2` chooses keys from the store to build up a message of the type `TExp2` to be sent to the card. Finally, the remaining transition just replays an expression stored in `exp2store`.

The intruder can also nondeterministically move to state `SimCard` where he simulates the card, in an analogous way to state `SimPSAM`.

Note that in this model, the intruder can only first act as the PSAM, and then as the Card. This restriction can be removed by adding a transition from `SimCard` to `SimPSAM`. The model can also be tailored, e.g. not to allow the intruder to act as the PSAM at all, or not to learn certain keys or intercept particular messages etc.

3.2 Model Checking

The system specification described above specifies the behaviour of the CEPS purchase transaction protocol in presence of a hostile intruder. This specification can now be simulated or tested. In this paper we concentrate on model checking to examine the protocol with respect to possible vulnerabilities.

We consider a vulnerability a behaviour which leads to the PSAM reaching the `OK` state without prior having received the transaction information *S* created by its *immediate* communication partner. Note that this is different from the situation of Internet protocols where communication is usually passed on by third parties (and possibly the adversary) due to the physical situation. Here, the holder of the card directly communicating with the PSAM receives the purchased goods without further authentication, which motivates the above definition. Since in our model the PSAM communicates directly only with the adversary, it is sufficient to check if it ever reaches the `OK` state.

The intruder first acts as the PSAM, only communicating with the card (state `SimPSAM`), and later as as the card (`SimCard`, only communicating with the PSAM). When there is an execution such that the PSAM reaches the state `OK`, the intruder managed to trick the PSAM into authenticating him.

For this purpose, we use the AUTOFOCUS connection to the model checker SMV. The property we check is $AG\neg(PSAM.State=OK)$, meaning

that in all reachable states (AG) the PSAM does not reach the stats `OK` in presence of the intruder.

If the property is violated, this indicates a vulnerability, and the model checker outputs a corresponding trace. Such a trace can then be automatically converted into an MSC and visualized in the tool AUTOFOCUS.

- Whether or not this situation arises depends on which keys the attacker possesses initially and how freely he can move between the states `SimCard` and `SimPSAM`. Below, we explain some scenarios we examined. Model checking of all scenarios took approximately 5 minutes on a SUN UltraSparc 2 with 1GB of memory and two 200MHz processors.
- If the intruder only possesses the public keys `KC`, `KM` and his private key `SKI`, the model checker does not find an attack. Thus, we can conclude that with respect to the chosen attacker model, the CEPS purchase transaction has been shown to be correct. Due to the restrictions of the model, this is no full proof of course — further evaluation of the protocol can be carried out later by more thorough methods as theorem proving (automatically or by hand).[2]
- Assuming the private keys `KC_` and `KM_` leaked somehow, so the intruder could get hold of them. Then of course he can authenticate himself to the PSAM. The model-checker correctly indicates this and outputs a corresponding MSC, which graphically visualizes the behaviour of the system in this case for the developer. Such *execution traces* can be generated by the model checker for many different kinds of specifications of possible runs (whether security related or not) and make it possible to test the implementation of the system or find and correct mistakes in the specification.
- Both private keys have to leak at the same time - if only one key leaks, the model checker does not find an attack. Thus, in the first threat scenario described in Section 2.2, no attack is possible without leaking keys.
- If one allows the intruder to move freely between the states `SimCard` and `SimPSAM` — as in the second threat scenario from Section 2.2 where the intruder communicates with the attacked card and the attacked PSAM in parallel — we find that the PSAM can actually reach the state `OK`. This is possible even if the intruder does not possess any keys. The corresponding execution trace is displayed in Figure 7 and shows the attacker acting as a relay, i.e. waiting for a message from the PSAM (in the `SimCard` mode), forwarding it to the card, waiting for the reply (in the `SimPSAM` mode) and forwarding it to the PSAM. The dashed lines

[2]However, these restrictions could be justified with arguments similar as done in (Lowe, 1996)

represent time ticks (each tick corresponds to the execution of a transition in the automata). We can see from the diagram that the intruder takes a number of ticks to record the messages before he can produce an answer. If the PSAM and the card wait when receiving messages, this does not restrict the model. In addition, there is a spurious fake message sent by the intruder, which is ignored by the card (this message was generated by the model checker as part of the counterexample — in our model the intruder can sent any fake messages at any time).

– If we rule out this kind of behaviour (e.g., by not allowing the intruder to replay both messages), again no attack can be found.

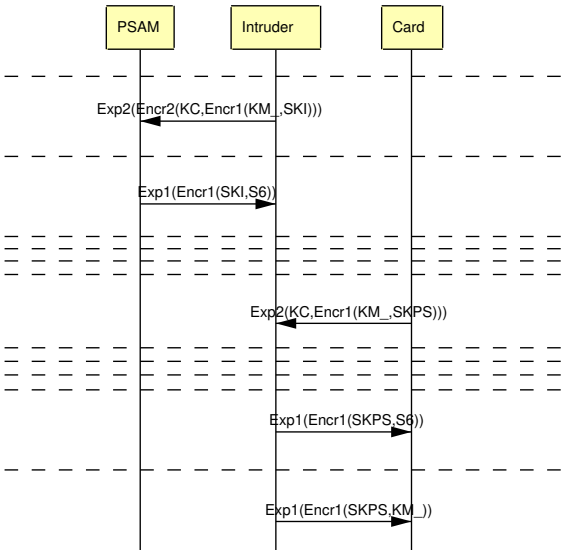

Figure 7. MSC for potential vulnerability in CEPS System

3.3 Interpretation of results

Our model showed that the CEPS purchase transaction is resistant to attacks of an intruder trying to have itself illegimately authenticated by the PSAM — except for the second, more sophisticated, threat scenario. As explained, this is no full proof of the correctness of the protocol.

The indicated potential vulnerability is present since firstly the card cannot communicate directly with the cardholder during the transaction and secondly CEPS does not include the entire POS device in the security perimeter. An intruder being able to attack a card and a PSAM in a remote place at the same time could therefore carry out purchase transactions with

the attacked PSAM on the account of the attacked card. Given the card specifications (no own display) to avoid the first cause one would have to employ a challenge and response between card and user via the terminal, as suggested in (Abadi et al., 1993), which may however not always be practical. So the right conclusion may be to increase security by securing the Card Acceptance Devices (CAD) of (unattended) POS devices so that communication with the PSAM is only possible with proper smart cards without contact to the outside of the POS device, and not possible by bypassing the CAD.

Note that the scenario given in Figure 7 would not at all be considered a vulnerability in the context of Internet protocols, where it is assumed that an attacker may act as a relay. This again shows the importance of considering the underlying physical situation when investigating security of systems (Gollmann, 2000). Note also that even though the scenario pointed out above does not require any cryptographic operations on the side of the adversary, our model does of course perform these (under the usual restrictions regarding key knowledge) in the general case. Of course, the message exchange in Figure 8 does not require use of model checking techniques to be found, however, one of the benefits of formal methods is that they require the investigator to be very explicit about assumptions made (on the underlying physical security), which may be why the scenario has gone unnoticed so far.

Future work includes a more formal justification showing that an infinite-state adversary is no more powerful than the used finite-state one by computing the transitive closure of the possible actions.

4. CONCLUSION AND FUTURE WORK

We investigated the security of the currently developed Common Electronic Purse Specifications (CEPS) using the distributed systems CASE tool AUTOFOCUS. Benefits of our approach include the possibility to specify and verify cryptographic protocols in the framework of a general CASE tool, which enables a treatment in the context of general system development. Since security violations often occur at the boundaries between security mechanisms (such as protocols) and the general system (Anderson, 2001), being able to treat protocols in the context of the system allows an adequate security assessment.

Apart from these methodological benefits, this work delivers concrete results on the security of the payment systems that are to be developed and fielded according to the CEPS. Our investigation exhibited a potential weakness arising from the fact that according to CEPS, the POS device is

not part of the security perimeter and especially from the intended future employment over the Internet. Due to space constraints we could only consider one part of the CEP specifications, other parts are left for further work.

Note that the protocol considered here is relatively simple, as our motivation is not to push the frontier of what is possible with model checking technology but to indicate how to incorporate formal techniques for security engineering into general system development.

This work consitutes only a very first step towards "computer-aided security engineering". We intend to go further beyond the scope of formal methods as it is previously mostly applied in computer security by considering vulnerabilities arising from the way security mechanisms are employed in the system context and by employing tools beyond model-checking and theorem-proving (such as specification-based testing which we recently applied to firewall design).

REFERENCES

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

N. Asokan, P. Janson, M. Steiner, and M. Waidner. The state of the art in electronic payment systems. *Advances in Computers*, 53, 2000.

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

R. Anderson. The formal verification of a payment system. In Mike Hinchey and Jonathan Bowen, editors, *Industrial-Strength Formal Methods in Practice*, pages 43–52. Springer, 1999.

R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989.

M. Broy, F. Dederich, C. Dendorfer, M. Fuchs, T. Gritzner, and R. Weber. The design of distributed systems - an introduction to FOCUS. Technical Report TUM-I9202, Technische Universität München, 1992.

A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *Computer Aided Verification*, LNCS. Springer, 2000.

CEPSCO. Common Electronic Purse Specifications, 2000. Business Requirements vers. 7.0, Functional Requirements vers. 6.3, Technical Specification vers.2.2, available from http://www.cepsco.com.

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

D. Gollmann. What do we mean by entity authentication ? In *IEEE Symposium on Security and Privacy*, 1996.

D. Gollmann. *Computer Security*. J. Wiley, 1999.

D. Gollmann. On the verification of cryptographic protocols - a tale of two committees. In *Workshop on Security Architectures and Information Flow*, volume 32 of *Electronical Notes in Theoretical Computer Science*, 2000.

Stefanos Gritzalis, Diomidis Spinellis, and Panagiotis Georgiadis. Security protocols over open networks and distributed systems: Formal methods for their analysis, design, and verification. *Computer Communications*, 22(8):695–707, 1999.

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.

F. Huber, S. Molterer, B. Schätz, O. Slotosch, and A. Vilbig. Traffic Lights – An AUTOFOCUS Case Study. In *1998 International Conference on Application of Concurrency to System Design*, pages 282–294. IEEE Computer Society, 1998.

ITU. ITU-TS Recommendation Z.120: Message Sequence Chart (MSC). ITU-TS, Geneva, 1996.

Jan Jürjens. Object-oriented modelling of audit security for smart-card payment schemes. In P. Paradinas, editor, *IFIP/SEC 2001 – 16th International Conference on Information Security*. Kluwer, 2001a.

Jan Jürjens. Secrecy-preserving refinement. In *Formal Methods Europe*, LNCS. Springer, 2001b.

Jan Jürjens. Towards development of secure systems using UML. In H. Hußmann, editor, *Fundamental Approaches to Software Engineering (FASE/ETAPS, International Conference)*, LNCS. Springer, 2001c.

I. Krüger. *Distributed System Design with Message Sequence Charts*. PhD thesis, Technische Universität München, 2000.

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

G. Lowe. Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR. In Margaria and Steffen, editors, *TACAS*, volume 1055 of *LNCS*, pages 147–166. Springer, 1996.

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

J. Philipps and O. Slotosch. The Quest for Correct Systems: Model Checking of Diagramms and Datatypes. In *Asia Pacific Software Engineering Conference 1999*, 1999.

Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its applications to secure message transmissions. In *IEEE Symposium on Security and Privacy*, 2001.

P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Analysis and Design of Security Protocols*. Addison Wesley, 2001.

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

O. Slotosch. Quest: Overview over the Project. In D. Hutter, W. Stephan, P Traverso, and M. Ullmann, editors, *Applied Formal Methods - FM-Trends 98*, pages 346–350. Springer LNCS 1641, 1998.

G. Wimmel, H. Lötzbeyer, A. Pretschner, and O. Slotosch. Specification Based Test Sequence Generation with Propositional Logic. *Journal on Software Testing Verification and Reliability*, 10, 2000.

G. Wimmel and A. Wißpeitner. Extended description techniques for security engineering. In P. Paradinas, editor, *IFIP/SEC 2001 – 16th International Conference on Information Security*. Kluwer, 2001.