The Use of Formal Methods for Trusted Digital Signature Devices

Bruno Langenstein and **Roland Vogt**
Deutsches Forschungszentrum für
Künstliche Intelligenz (DFKI GmbH)
[German Research Center for Artificial Intelligence]
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{langenstein,vogt}@dfki.de

Markus Ullmann
Bundesamt für Sicherheit in
der Informationstechnik (BSI)
[German Information Security Agency]
Godesberger Allee 183, 53133 Bonn, Germany
ullmann@bsi.de

Abstract

This paper presents a formal security policy model for SmartCards with digital signature application. This kind of model is necessary for each evaluation according to Information Technology Security Evaluation Criteria assurance level E4 (Common Criteria level EAL5) and above. Furthermore, we argue that such a model is essential for reasoning about the security of Information Technology components like a specific IT product or IT system. Without an unambiguous definition of what security means, it is impossible to say whether a product really is secure.

Introduction

In 1997 the government of the Federal Republic of Germany established the Information and Communication Services Act. It contains the Digital Signature Act [1] as article 3. On top of that law the German parliament approved the Digital Signature Ordinance [2]. The aim of [1] and [2] is the definition of requirements that enable the equivalence of manual and digital signatures concerning legal binding, authenticity and integrity of electronic documents in the future. One specific requirement of [2] (cf. § 17) is the evaluation of technical components according to the Information Technology Security Evaluation Criteria [8]. Especially, the devices for the generation of cryptographic keys as well as the facilities for storing and applying the private signature key require an evaluation according to assurance level E4 of [8]. A formal model of security policy (FMSP) is needed to reach this level.

SmartCards are appropriate devices for the generation of digital signatures. The German standardization organization (DIN) developed an interface specification for SmartCards with signature application/function [4]. The aim of this standard is to guarantee inter-operability. It specifies the interface between a terminal (interface device) and a digital signature card which is in compliance with the German digital signature law. This specification takes all the legal regulations into account. It is based on the ISO-Standard 7816 and defines the data objects within a file structure as well as a collection of interface commands to be used for the generation resp. validation of digital signatures.

The generic formal model of security policy presented here has been developed within a pre-evaluation project of SmartCards with signature application. It is based on the interface specification [4] and defines the technical security policy in correspondence with the accompanying generic security target (GST) [9]. The target of evaluation is the embedded software of the SmartCard exclusively. Neither the hardware nor the card operating system is subject to closer examination. In the following parts we use the term 'SigCard' as an abbreviation for the target of evaluation.

SigCards and Assumed Threats

In our context we look at SmartCards which are ready for signing. This means that the manufacturing and personalization processes of the SmartCard are completed, i. e. the integrated circuit of the SmartCard is loaded with the digital signature application and the data base related to a specific cardholder. It is assumed that this data base has been processed by the personalization authority in a secure way and that the SmartCard is securely handed over to the legitimate cardholder.

The digital signature operation is processed in the SigCard. For that process the secret signature key of the cardholder is used which is stored in a secure manner within the SigCard. The SigCard is accessed with a SmartCard reader connected to e. g. a personal computer which is responsible for transmitting the data to be digitally signed. In the following we denote all external devices (card readers, terminals, ...) which might be involved in accessing the SigCard with the term 'InterFace Device' (IFD).

The German signature legislative distinguishes 'private' and 'public' IFDs. The cardholder is expected to trust his own private IFD. Here the cardholder himself is responsible to use an IFD which is compliant with the legislative. Public IFDs provide commercial services. In this case the cardholder is not able to control or even know whether the IFD is compliant with SigG legislative.

The idea of the security policy of the SigCard is that the cardholder is the only legitimate user. Not any other person is allowed to use her/his SigCard. The SigCard contains the cardholder's secret key for signing. It is regarded as the most valuable asset. The SigCard must keep this key confidential at all events. Digital signatures are only produced with the explicit permission of the cardholder.

In preparation of the development of the FMSP a threat analysis has been performed. That analysis didn't take the hardware or the operating system of the SmartCard into account. The examination was restricted to the embedded application software and data of the SigCard and revealed the following threats:

T1 Extraction of the secret signature key from the SigCard by directly reading the data or by analyzing computational results.

T2 Using the digital signature application of the SigCard without having the permission of the cardholder.

T3 Unnoticeable modification of signed data.

T4 Presentation of a forged SigCard to a public IFD without the public IFD being able to notice that.

T5 Presentation of the SigCard to a forged public IFD without the cardholder being able to notice that.

The specific threats can be summarized as such: The cardholder's signature is generated for a piece of data the cardholder does not want to sign.

Formal Model of Security Policy

This section describes the formal specification defining the security policy of the SigCard. It formalizes the important security principles and relates them to the security objectives described in the generic security target [9]. The formal model was developed using the 'Verification Support Environment' (VSE) which is a tool supporting formal methods in the complete software life cycle [7]. It is officially approved for the definition of formal models as required by [8] or [3]. Due to the limited amount of space we only can give a brief sketch of the FMSP in the present paper. The complete formal specification together with its informal interpretation and other related documents are available in electronic form [5].

Design Principles

The interface of the SigCard is modelled by input and output variables. The regular input and output of the SigCard is modelled by an input variable `channelIn` and an output variable `channelOut`. The values of these variables abstractly represent the incoming resp. outgoing messages to resp. from the SmartCard.

In order to be able to distinguish between a message that is sent once and one that is sent several times a gap is modelled which separates two messages. This is done using the value `noInfo` representing the absence of information. We expect the input variable `channelIn` to change from `noInfo` to a value different from `noInfo` and back again, but never to immediately change between two values both different from `noInfo`. The SigCard is modelled in such a way that this is also valid for the output variable `channelOut`.

According to ISO 7816 the communication between the SmartCard and the terminal is like a question and answer game. The SmartCard answers to commands received from the IFD. If the IFD does not see an answer within a well defined period of time it may try to send the command again or (after a few trials) reset the card. Commands and answers are both modelled as elements of type information that will also be called messages in the following.

Since the card is the slave part in the communication it is the responsibility of the IFD to react if an expected answer is not returned. Therefore, we need not put any timing constraints on reactions and answers from the card. It is not necessary that a command is reacted upon by immediately producing a corresponding answer. It suffices to require that if the SigCard reacts to a command, the answer properly reflects its internal state.

Nevertheless, as we are concerned with the flow of information between the SigCard and its environment we have to take temporal aspects into account. So the main part of the specification is formulated in a temporal logic specification language [6]. In order to define the states which may change as time proceeds we need data elements that form these states as well as static relations and functions on these elements. This leads to an underlying structure providing abstract data types necessary for our specification.

The most important abstract data types are `subject` and `information`. Elements of type `subject` are equipped with a certain amount of knowledge consisting of an undetermined number of elements of type `information`. Subjects can enrich their knowledge base by learning new information. This process doesn't change the identity of the resp. subject. The main purpose for introducing these data types is the definition of the predicate `inferable` which describes the principal possibility of information extraction. According to the defining axiom `inferable(i,k)` is valid if there is at least one subject which knows `k` after learning `i` although it did not knew `k` before.

The formal specification is divided in two layers. The upper layer takes a global viewpoint and describes the SigCard together with the environment in which it is expected to be used. It formalizes the security objectives specified in [9] as well as assumptions about the environment. We call this part of the specification the 'security objectives layer'.

The lower layer takes a local viewpoint and describes the SigCard at an operational level. It defines the security policy of the SigCard in terms of a formalization of the important security principles extracted from [9]. We call this part of the specification the 'security policy layer'.

The two layers are connected by a `satisfies` link. It expresses that the operational behaviour specified in the policy layer should fulfil the properties formalized in the objectives layer. More generally such a link states that every correct implementation of an operational specification should be a model (w. r. t. the semantics of the specification language) of the associated requirement specification. In our case the operational specification corresponds to the policy layer while the requirement specification corresponds to the objectives layer.

The `satisfies` link causes the VSE tool to automatically generate proof obligations. If they can be proved the satisfies relation is valid. In the present context this means that we have a formal proof that the security policy meets the security objectives.

Security Objectives Layer

On the upper layer the SigCard is described according to the security objectives specified in [9]. We will present here some of these objectives together with their appearance in (the objectives layer of) the FMSP. To each security objective one or more temporal logic formulas are associated.

SO1.1:

```
[] NOT inferable(channelOut,skCh)
```

SO1.2:

```
[] skCh = skCh'
```

SO2.1:

```
[] ALL i:
   inferable(channelOut,sig(i,skCh))
   ->  authUser = t
```

SO2.2:

```
[] maxFailuresExceeded(channelOutHistory)
   -> authUser = f
```

SO2.3:

```
[] ALL i:
   inferable(channelOut,i)
   -> NOT authCh(i)
```

SO7.1:

```
[] ALL i:
   inferable(channelOut,sig(i,k))
   -> command = doSign(i) AND
      skCh = k
```

Figure 1: Excerpt from Security Objectives Layer

The security objective **SO1** in [9] reads as follows:

The SigCard ensures the confidentiality and the integrity of the private signature key of the cardholder stored in the SigCard.

This security objective counters threat **T1**. It is modelled by several formulas (cf. Fig. 1) concerning the state variable `skCh` that represents the cardholder's secret key.

SO1.1 Extracting the cardholder's secret key is only possible if there is an output from the card that allows anybody to infer the signature key. Therefore, the above formula requires that at any moment the secret key `skCh` cannot be inferred from the output of the SigCard. Note, that this statement requires some assumptions on the input stream. It can only be valid if we assume that the secret key was not previously transmitted to the SigCard (e. g. as a piece of information to be signed).

SO1.2 This statement requires that at any time the value of the cardholder's secret key in the present state is equal to the value of the secret key in the next state. So the secret key can never change.

The security objective **SO2** in [9] reads as follows:

The SigCard shall allow the use of the digital signature function only to the cardholder.

This security objective counters threat **T2**. It is modelled by several formulas (cf. Fig. 1) concerning the state variable `authUser` that reflects whether the user currently is authenticated as the cardholder and the state variables representing the cardholder's authentication data.

SO2.1 The application of the signature function is observable if a message appears in the output of the SigCard in any form that allows the signed piece of information to be inferred. Therefore, the formal model requires that `authUser` must indicate that the user is authenticated whenever a signed piece of information can be inferred from the output. Of course, the specification contains a number of additional formulas which ensure that the variable `authUser` really reflects the correct authentication state.

SO2.2 Successive authentication failures will be interpreted as an attempted unauthorized access by the SigCard and should disable the signature application. This is formally modelled by requiring that the user must not be authenticated, i. e. `authUser` equals `f`, whenever the maximum number of successive authentication failures is exceeded.

SO2.3 An important condition for an effective cardholder authentication is that the relevant information, usually the PIN, is kept secret. Therefore, we formally require that every piece of information `i` inferable from `channelOut` is different from the cardholder authentication data. Again, this statement can only be valid, if the authentication data was not previously transmitted to the SigCard.

The security objective **SO7.1** in [9] is the principal[1] security objective for SigCard's and reads as follows:

The SigCard provides a function to generate a SigG digital signature for the data presented by the IFD using the SigG private signature key of the cardholder stored in the SigCard.

This security objective counters threat **T2**. It is modelled by a formula (cf. Fig. 1) concerning the state variables `command` that represents the command currently processed by the SigCard and `skCh` that represents the cardholder's secret key.

SO7.1 Whenever a piece of information `i` signed with key `k` can be inferred from the output of the SigCard then it should have received a request for signing this data and the signature key used should be cardholder's secret key. Note, that it makes no sense to require that every signing request should eventually be answered with a resp. signature due to the restrictions imposed by user authentication.

[1] Within the FMSP we don't take the second part **SO7.2** into account since it expresses requirements on cryptographic algorithms that have no impact on the security policy of the SigCard's application software.

There are several additional security objectives specified in [9]. They are concerned with requirements for public IFD's resp. with the reaction to security violations. All of these security objectives are covered by collections of formulas similar to those described above. We end the presentation of the objectives layer with the remark that it does not contain any restriction on how the implementation of the signature application should meet the security objectives.

Security Policy Layer

The specification of the security policy in [9] suggests a decomposition of the specification into components according to the policy. Therefore the policy layer of the FMSP is a combined temporal logic specification. This means it consists of components that work in pseudo parallel manner. In every state one of the components is active and can perform its state transitions while the other components perform 'stuttering' steps[2]. Fairness conditions are in force ensuring that every component will eventually take a step provided the preconditions of the specification hold. A component is said to be enabled if its preconditions hold.

The individual components of the policy layer are described in the following paragraphs. The description includes the most important aspects of the control and data flow within the policy layer.

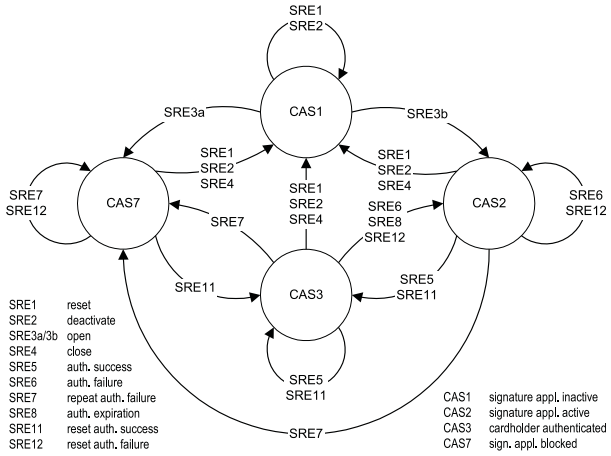

Figure 2: Excerpt from Security Policy Layer

`Automaton` component The component `Automaton` is responsible to maintain the current authentication state (CAS). With the help of the object components it detects security related events (SRE) and reacts on them by changing the state accordingly. This state machine defines the important security principles of the SigCard and is depicted in Fig. 2. Due to space limitations the illustration only includes the 'private IFD' part of the state machine.

Furthermore the `Automaton` component performs the task of a scheduler. Actions which have to wait because they depend on the results of other components are disabled until the `Automaton` passes control to the resp. component. Each time such an action is finished the control is returned back to the `Automaton` component which in turn decides how to proceed.

Object components The state of the formal specification is distributed to a number of object components, each representing an object[3] as described in [9]. An object contains an asset which is protected against unprivileged access. Among these values are

- the secret key for producing digital signatures,
- the cardholder authentication data,
- data needed for signature verification,
- mutual device authentication data, and
- data related to the communication between SigCard and IFD.

The asset of an object is stored in an internal variable of the component. This way it cannot be accessed directly. Each access operation is modelled as a distinguished action. The action is enabled according to the access rights based on the current authentication state. Each object component has an output channel representing the output of the SigCard generated by that object. This can be the value stored in the object or some other value computed using the stored value.

Components for Secure Communication Input and output of the SigCard never enter or leave the object components or the automaton component directly. All input and output passes through the `SecChannelIn` resp. `SecChannelOut` component. These components are responsible for secure messaging. Outgoing messages are signed and encrypted, incoming messages are decrypted and their signatures checked. If secure messaging is not in force (before device authentication) then these components only forward input and output values.

The `SecChannelIn` component performs a loop containing two actions. The first action simply waits until the input variable `channelIn` contains the value `noInfo`. Input is only read thereafter when `channelIn` has changed to a value indicating a regular piece of information. This prevents the same input from being processed twice. At the end of each loop the `SecChannelIn` component (automatically) stutters allowing the other components to work on the input and produce appropriate output.

The `SecChannelOut` component performs the complementary loop consisting of an operation which passes the output produced by an object or the `Automaton` component to `channelOut` followed by an action which sets it back to `noInfo`. Both loops run in parallel with each other and with the main processing loop specified in the `Automaton` component.

[2]The transactions of a temporal logic specification are arbitrarily interleaved with so called stuttering steps which randomly change state variables that are allowed to do so (e. g. input variables).

[3]The digital signature application itself is referenced as an object in [GST]. This is not reflected in the FMSP since such self referencing constructions cause some problems in formal specifications.

Results and Benefits

The generic security target [9] and the accompanying FMSP for SigCards were developed in a parallel manner. The formalization process of the security policy started with a first draft of the GST describing only some parts of the SigCard's security features. The description was completed step by step. Each extension of the GST caused a new analysis trying to include the additional features in a formal style within our model. A lot of inconsistencies and contradictions in the GST drafts were discovered during the analysis and formalization process.

This is a general observation from all other formalization projects which we go ahead with. This means that the must to formalize a specific security behaviour alone accomplishes a specific security advantage. Always small modifications may cause serious errors which are not recognized if specified in natural language only.

In the present case a minor modification introduced in the almost final GST document caused a serious security problem. The security policy described in this late release of the GST always allowed to pass from `CAS1` to `CAS2` (cf. Fig. 2) via `SRE3` which was defined to unconditionally activate the signature application. We observed that `SO2.2` (cf. Fig. 1) was not provable. Indeed, the reason is that it was not valid, because at least one authentication attempt was allowed in `CAS2`. Implementing the SigCard application according to such a security policy would allow a potential attacker to brake the cardholder authentication by a brute force attack. Finally, the formally verified security policy replaces the former `SRE3` by two events as presented in Fig. 2. This example impressively demonstrates the benefits of formal methods since although the actual design flaw can easily be explained it remained undetected until the formal verification attempt failed.

Furthermore a potential vulnerability in the security policy of the standardized SmartCard was detected. According the law a cardholder can calculate his signature with his SmartCard connected to his private/company interface device or a 'public customer service terminal'. In the case of a 'public customer service terminal' it was detected that the cardholder himself has a large responsibility in analyzing whether the 'public customer service terminal' really is a trusted device *before* trying to authenticate himself.

The formal model of security policy defines the essential security behaviour of the SigCard and necessary assumptions on the environment in an abstract description as explained in the previous chapter. In such a situation it is possible to gain a complete survey of the security specific concepts and their interactions in one context. Once the security objectives based on the security policy layer have been formally verified then a complete knowledge about the security interdependencies on that level of abstraction exists. Furthermore, the benefit for the evaluation task is that the evaluator can check the real IT product very effectively against the formally specified security behaviour.

Today, the whole power of FMSP's is not completely understood. In most cases the development of a FMSP is performed only for evaluation purposes because the current security evaluation criteria like ITSEC or CC demand such a FMSP. Normally, the whole evaluation work is done after the development of the product. This includes the construction of the model, too. Looking a little bit deeper into the model it is obvious that the security policy layer could be an excellent requirement specification for the implementation of the security functionality of the product. Using it in such a manner helps to develop the product in a way that its implementation meets the security objectives. The consequence of this observation should be to use formal methods already in the early design phases of a product giving good assistance to the system designer. Today, we give away this very cost-effective advantage of the formal modelling by performing it only for isolated evaluation purposes. One reason for this is that the system designers are not familiar with formal specification languages based on logics like VSE-SL, necessary for the formalization work.

References

[1] Gesetz zur digitalen Signatur (Signaturgesetz – SigG). Entered into force as from August 1st, 1997. Available as 'Digital Signature Act' from `http://www.iid.de/iukdg/iukdge.html#3`.

[2] Verordnung zur digitalen Signatur (Signaturverordnung – SigV). Entered into force as from November 1st, 1997. Available as 'Digital Signature Ordinance' from `http://www.iid.de/iukdg/sigve.html`.

[3] *Common Criteria for Information Technology Security Evaluation (CC), Version 2.0*, 1998. Available from `http://csrc.ncsl.nist.gov/nistpubs/cc/`.

[4] *DIN V 66391-1: Spezifikation der Schnittstelle zu Chipkarten mit Digitaler Signatur-Anwendung / Funktion nach SigG und SigV*, December 1998. Available as 'Specification of chipcard interface with digital signature application / function according to SigG and SigV' from `http://www.bsi.de/aufgaben/projekte/pbdigsig/`.

[5] Bundesamt für Sicherheits in der Informationstechnik (BSI) [German Information Security Agency]. *ICC embedded software for Signature Creation conforming with German SigG, SigV and DIN V 66391-1 — Generic Formal Model of Security Policy and its Informal Interpretation in terms of the Generic Security Target*, March 2000. Available from `http://www.dfki.de/fmg/projects/fmdin`.

[6] D. Hutter, H. Mantel, G. Rock, W. Stephan, A. Wolpers, M. Balser, W. Reif, G. Schellhorn, and K. Stenzel. VSE: Controlling the Complexity in Formal Software Development. In *Proceedings Current Trends in Applied Formal Methods, FM-Trends 98*, volume 1641 of *Springer LNCS*, Boppard, Germany, 1999.

[7] F. Koob, M. Ullmann, and S. Wittmann. The New Topicality of Using Formal Models of Security Policy within the Security Engineering Process. In *Proceedings Current Trends in Applied Formal Methods, FM-Trends 98*, volume 1641 of *Springer LNCS*, Boppard, Germany, 1999.

[8] Office for Official Publications of the European Communities. *Information Technology Security Evaluation Criteria (ITSEC), Version 1.2*, 1991.

[9] TeleTrusT Deutschland e. V. *Generic Security Target for ICC embedded software compliant with German SigG, SigV and DIN V 66391-1*, March 2000. Available from `http://www.dfki.de/fmg/projects/fmdin`.