

An Integrated Framework for Security Protocol Analysis

Marcin Olszewski
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052, USA
marcino@microsoft.com

Lukasz Cyra
Gdansk University of Technology
11/12 Gabriela Narutowicza Street
80-952 Gdansk-Wrzeszcz, Poland
lukasz.cyra@eti.pg.gda.pl

ABSTRACT

Assurance of security protocols needs particular attention. Flaws in a protocol can devastate security of the applications that rely on it. Analysis of the protocols is difficult and it is recommended that formal methods are employed to provide for higher levels of assurance. However, the formal methods can cover only a part of the scope of the problem. It is important that the formal models are valid representations of the protocol and that the application context is adequately represented. In the paper we present an analytical framework that integrates the object-oriented and formal modeling approaches. Object models are used to capture the relevant aspects of the protocol and its security context and to communicate with the protocol designers. Formal models are applied to verify the protocol security properties. Applicability of the framework was demonstrated by several industrial case studies.

Categories and Subject Descriptors

C.2.2 [Computer-Communication Networks]: Network protocols - *protocol verification*; F.4.3 [Mathematical Logic and Formal Languages]: Formal Languages

General Terms

Documentation, Design, Security, Verification.

Keywords

analytical framework, object orientation

1. INTRODUCTION

Security protocols are subjected to subtle design errors that are difficult to analyze. Assuring such protocols requires application of advanced analytical methods and is commonly perceived as a niche where formal methods can be successfully applied. In recent years there were numerous attempts to apply formal methods to security protocols analysis [1].

Successful application of formal methods in the analysis of security protocols faces several limitations. Verification of protocol properties implies complex computations which can

easily exceed the available resources. In many cases the practically significant protocols are too complex to be efficiently analyzed. This complexity can be sometimes managed by applying abstraction and decomposition principles.

A formal model is always a simplification of the real protocol and is based on numerous assumptions [2]. To achieve the traceability of modeling decisions and assumptions and to be able to relate the results of formal analysis to original designs, the analysis should be carried out within a proper framework. Such a framework should support the informal-to-formal transformation of the protocol specification, identification and documentation of the underlying assumptions and clear and complete presentation of the results of the analyses. It should also facilitate considering the protocol in its operational environment and monitoring the validity of the underlying assumptions. The framework should be flexible enough to accommodate different formal techniques and benefit from their diversity.

We propose an integrated framework for security protocol analysis which combines formal modeling techniques for cryptographic protocol verification with object-oriented analysis in a well-defined engineering process. The framework ‘forces’ its user to document the results of the analyses and to identify and document all the underlying assumptions. Semiformal object-oriented modeling combined with formal modeling and analysis complement each other making use of their strengths: comprehensibility and versatility of object models and precision, unambiguity and rigor of the formalism.

In the subsequent sections we first overview the related works, then briefly describe the proposed framework and demonstrate its applicability in the context of two case studies. In conclusions we summarize our contribution and present plans for future research.

2. RELATED WORK

To date, numerous formal approaches to security protocol analysis have been proposed [1, 3]. Main differences are in the general modeling approach taken by a method and in the ways of performing the analyses. The latter distinguishes between theorem proving and model checking. Theorem proving includes techniques based on specialized logics, such as BAN [4], but also methods like Paulson’s inductive approach [5] or those relying on abstract algebras and morphisms [3]. Model checking approaches include methods created specifically for the security domain, like AVISPA [6] or strand-spaces of Athena [7], but also approaches derived or adapted from already established formal frameworks, like CSP process algebra [8], FOCUS [9] or spi-calculus [10].

In our work we concentrate on model checking techniques use them a ‘building blocks’ of the proposed framework. Those

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ASIACCS’08, March 18-20, Tokyo, Japan

Copyright 2008 ACM 978-1-59593-979-1/08/0003 ...\$5.00.

blocks are embedded in the UML based context which provides support to the modeling task before the actual formalization of the protocol. In practice, such modeling is performed most times very informally. It is supported by textual descriptions and some diagrams, or the so-called „standard notation” – with Alice and Bob exchanging messages [11]. Our contribution is by proposing UML as a main mechanism to support this ‘before formalization’ modeling task.

UML models of a protocol have to be eventually transformed into formal models. In general, the problem of refinement of UML models into more precise notations and languages or even executable code is a complex [12, 13] and still a subject of active research in the field of software engineering. Some recent results are based on the concept of viewpoints and unification [14]. A viewpoint is a partial model of a system, prepared in an object-oriented notation, such as UML. Unification is a process of combining partial models into a complete formal specification which may become further refined towards executable code.

Modeling security-critical systems and protocols in UML is not a new idea. In [15] Jurjens proposes UMLsec – a UML dialect for modeling heterogenous systems and their security properties including communication activities. UMLsec has a wide scope and is a powerful modeling tool. Models are prepared with the purpose of formalization and analysis, including code generation.

Our modeling approach is more focused than UMLsec. It uses simpler and fewer modeling constructs. Our approach is pattern-oriented. By using patterns we make the UML modeling more streamlined by clearly defining the focus of analysis. UMLsec has a far larger scope and addresses many issues outside the security protocol analysis. By focusing our framework on security protocols we managed to obtain a tool which is simpler but at the same time covers some areas specific to protocols which were neglected in [15], such as modeling assumptions or constraints resulting from formalization.

Integration of UML and formal methods for the purpose of cryptographic protocols specification, design and analysis was studied in the CASENET project [16]. CASENET delivered an integrated approach encompassing the entire lifecycle of a protocol – from gathering of functional requirements through protocol specification and design to implementation (or rather, construction from modular components), validation and testing. UML diagrams are used to capture requirements and model the dynamics of protocols, the notation is assisted by a SRL formal language. A suit of advanced applications support the CASENET methodology, including a powerful commercial validation tool SAFIRE. Our framework is more light weight than the CASENET approach. It focuses on protocol modeling and analysis and on traceability of the analysis process and communicating the results to the stakeholders. In particular, protocol design and implementation fall outside the scope of our framework. Protocol modeling and specification in CASENET are very tightly integrated with the supporting tools which can be considered an advantage in terms of automation and scalability. Our framework assumes a loose coupling with formal methods and supporting tools, which gives more flexibility and helps in exploiting an additional potential resulting from their diversity. In the sequel we report on our experience with using different formalisms: CSP/Casper, FOCUS and AVISPA.

3. FRAMEWORK DESCRIPTION

3.1 Framework architecture

The framework integrates various methods and tools with the intention of supporting the user while carrying different phases of protocol analysis, starting from an informal, ‘technical’ formulation of the protocol and its environment, through modeling the protocol and its context using semiformal, graphical notations and tools and ending at a formal model and its precise, mathematically founded analyses.

The main components of the framework are illustrated in Figure 1.

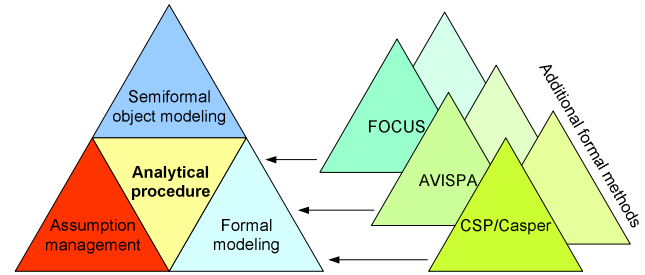


Figure 1. Components of the integrated framework for protocol analysis

The heart of the framework is the *analytical procedure* which defines a workflow to be followed while applying the framework. Here is an overview of the procedure steps:

Analysis of input material – Elicitation of facts relevant to protocol modeling and analysis based on documentation such as protocol specifications, design documents, technical reports but also taking into account interviews with designers, meeting reports etc.

Identification of assumptions and simplifications – Identification and documentation of assumptions related to the protocol and its environment. It also includes identification and documentation of all simplifications necessary for making the modeling and formal analysis feasible.

Identification of security goals – Identification and specification of the security objectives of the protocol (and all its subprotocols) – verifying whether the protocol actually satisfies these goals is the purpose of formal analysis.

Validation of analysis scope – Validation of the results of the previous steps (with the help of domain experts and protocol designers).

Semiformal modeling and analysis – Developing UML object models that represent protocol participants and their interactions (protocol dynamics), the threat model, and documenting in the models all the security objectives and underlying assumptions.

Formal modeling and analysis – Developing formal model of the protocol; verification of the formal model against the security objectives of the protocol. Formal modeling and analysis can be carried out using one of the following methods and tools: Casper/CSP, FOCUS and AVISPA.

Documenting the results – The results of the analysis are documented using the predefined template.

3.2 UML modeling

The object modeling component of the framework is based on UML [17]. The language was extended by stereotypes proposed for common elements of protocols models, for example <<protocol agent>>, <<session>> and <<secure message>> which were derived respectively from UML actor, object state and object interaction. The next step towards this direction is a complete domain-specific UML profile [18] which is in line with the OMG recommendation for those UML users who want unambiguity and support for tool automation [19]. UML 2.0 offers a revised profiling mechanism. Stereotypes are introduced by means of inheritance from a standard UML metaclass, and domain-specific features of the new stereotypes are expressed formally using the Object Constraint Language (OCL).

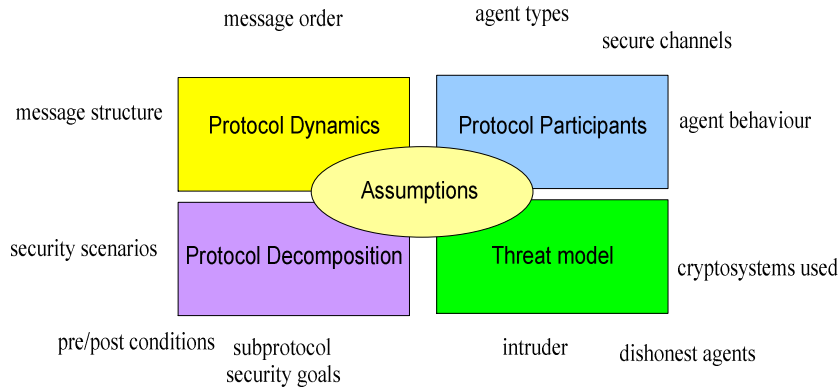


Figure 2. Security protocol modeling perspectives

To help the users in using the proposed UML extension we provide a set of patterns. Each pattern defines a scheme for constructing a particular aspect of the protocol model from a specific perspective. While defining the patterns took into account the needs of formalization of the protocols. For instance, all three formal techniques we have experimented with: CSP/Casper, AVISPA and FOCUS required the following aspects to be included in the specification:

protocol participants (agents) – agent behavior in response to received data, expectations towards security services offered by the protocol, properties of channels used to send and receive messages

protocol dynamics - the way messages are constructed from variables and sent between agents

threat model - capabilities and knowledge of the intruder

protocol decomposition – partitioning a protocol into smaller subprotocols to limit the scope, size and structure of the network running the protocol, declaration of protocol sessions for agents to run in a given scenario

We have identified a set of perspectives which forms a common modeling baseline for formal security protocol analysis. The perspectives are shown in Figure 2. The “Assumptions” aspect is put in the central place, as we recognize that identification and documentation of assumptions should be the central theme of protocol modeling regardless of the perspective.

The modeling patterns serve a number of purposes. Firstly, they help to focus on one particular problem at a given stage.

Secondly, they help the analyst to better understand the way each concept is used not by studying the OCL constraints but by observing how this concept behaves in relation to other concepts. The patterns are defined in accordance with the following template:

Pattern name – a descriptive name of the pattern

Intent – which modeling issues are addressed by the pattern

Definition - UML diagram defining the pattern; description of the pattern and its elements

How it works – explanation of how the pattern works in practice; supported by a real example from one of the Case Studies

Reference to other patterns – as patterns are used together to

model different aspects of the protocol, this section explains the relationships between this pattern and the others

Variants - because of a few differences of how certain aspects of the protocol are modeled in AVISPA, CSP/Casper and FOCUS, several patterns have more than one variant tailored to suit specific requirements of a corresponding formalism

The patterns impose some structure on the UML models which is then reflected in formal models. This has certain advantages, which will be explained in the following sections.

Example pattern definitions are shown in Figure 3 and Figure 4.

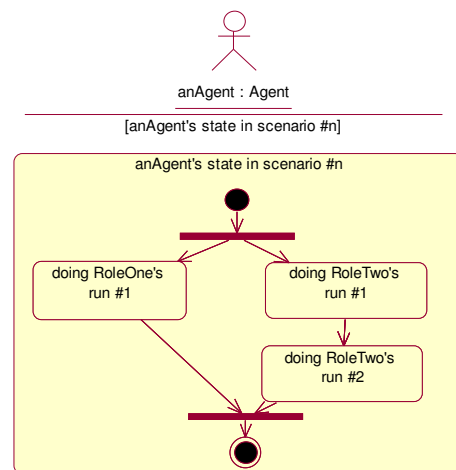


Figure 3. An Agent's sessions pattern definition – CSP variant

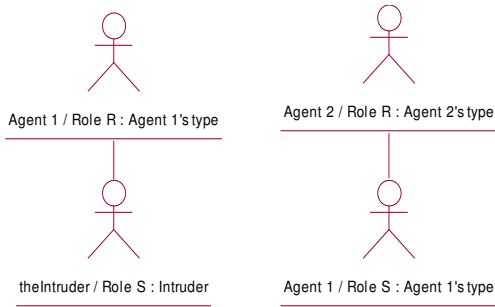


Figure 4. An Agent's sessions pattern definition – AVISPA variant

The pattern specifies possible behaviors of an agent. In CSP variant of the pattern (see Figure 3) each agent is characterized by a single compound state which contains sub-states representing individual sessions the agent can run. The agent can assume multiple roles and run the sessions concurrently or consecutively.

Even though there are strong similarities and significant overlap between different formal approaches to protocol modeling and analysis, there are also a few important differences which must be taken into account. For example, the “Agent's sessions” pattern represents a specific facet of CSP/Casper. In this approach the network running the protocol (also called a system) is described by specifying a number and sequence of sessions each agent can run and what roles he can take in each session. What is important is that the analyst does not specify what other agents will collaborate in each session. The model checker tries all possibilities by testing each possible combination of agent identities, provided that those other agents are allowed to run a session taking a complementary role.

The AVISPA and FOCUS methods take a different approach. The analyst assigns sessions and roles to each agent in the system but he also explicitly says which agents are running each session. Different combinations require either a more complex model with more sessions (which may quickly become too large to process) or one can model a separate security scenario with its own script. To capture this property of AVISPA/FOCUS, we proposed more than one variant of the Agent's sessions pattern. While Figure 3 shows the CSP variant of the pattern, Figure 4 presents the version intended for AVISPA/FOCUS users. In this case the model takes the form of a collaboration diagram depicting instances of different types of agents with object links representing the fact that given agents are involved in a protocol session and are taking roles specified in collaboration.

Note that with this variant the intruder has to be explicitly mentioned in a model if he should participate in a protocol session posing as a legitimate agent. That is just another difference between AVISPA or FOCUS and CSP/Casper. The former method allows the intruder to run an arbitrary number of sessions without explicitly specifying this fact in the threat model.

3.3 Assumptions

Modeling inevitably involves making assumptions. Understanding those assumptions and their implications is crucial for the analyses and the interpretation of the results. The framework supports assumption management in two ways: the metamodel helps in identifying the areas that should be considered for possible assumptions and the analytical patterns provide means

for documenting the assumptions. We distinguish different types of assumptions, including: scope of analysis, security of cryptographic mechanisms, scope of threats and protocol simplification. The assumption documentation template is given in Table 1.

Table 1. Assumption documentation template

ID	Unique assumption identifier
Type	Assumption type
Body	Assumption expression
Motivation	Explanation why the assumption is needed
Rationale	Justification why this assumption is valid
Impact	How this assumption influences the analysis process and its results
Comments	Other relevant information

In addition to the above textual documentation, assumptions are also documented in object diagrams using stereotyped nodes <<assumption>>, which point to the assumptions documentation by referring to their ID, as illustrated in Figure 5.

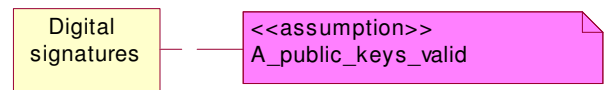


Figure 5. Documenting assumptions

3.4 Formal analysis

To provide for unambiguity of specifications and precision of the analyses the framework employs formal modeling. Formal models are derived from the object models during the formalization step of the analytical procedure. By integrating more than one formal method into our framework, we can benefit from added diversity at the same time reusing most of the work needed to prepare semi-formal models and identify and document the necessary assumptions. At present, the framework supports three protocol analysis techniques based on model checking: CSP/FDR, AVISPA and FOCUS.

Formal specifications are prepared manually by applying some formalization rules and guidelines. Formalization rules refer to entire patterns or their larger fragments. This approach was inspired by the techniques of model formalization based on viewpoints [14, 13]. Our formalization rules and guidelines follow the ideas of model unification (but in a less formal way). Unification recognizes viewpoint consistency as a fundamental issue (inconsistent models cannot be unified to a well-formed formal specification). In our approach this problem is somewhat lessened, if the analyst follows the modeling patterns accurately. The rules leave some room for interpretation and therefore we do not claim to have a rigorous method of semi-formal model refinement. Our formalization rules meet the requirements postulated in [12]: (1) their scope goes beyond basic model elements to entire patterns; (2) they are documented in a natural language, same as the patterns themselves; (3) the models link individual assumptions to specific model elements which provides for greater traceability.

Figure 6 shows an example procedure of translating a model using the Agent Sessions pattern into Casper instructions. Casper scripts are then automatically compiled into machine readable dialect of CSP and verified using the FDR model checker.

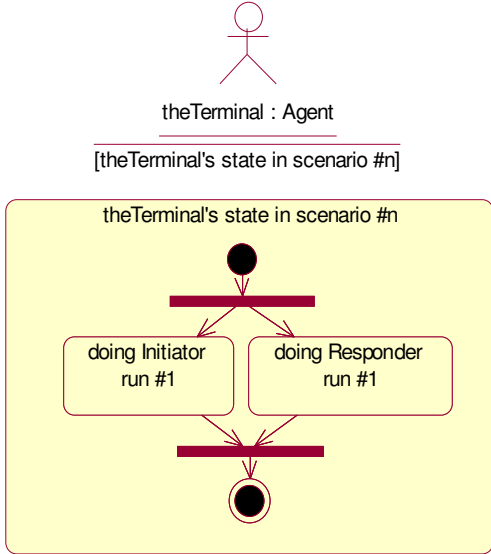
Model element	Sessions performed by agents
UML	Casper formal specification
Protocol sessions which an agent can run in a given security scenario are modeled using a state model. Each agent object is assigned a compound state which consists of several atomic states, each representing one protocol run. Individual atomic states can be either consecutive or parallel.	Individual runs an agent can run are represented as instances of CSP processes. Each type of process corresponds to a single role an agent can assume. Atomic UML states comprising agent's compound state are mapped into process instances, which are parameterized using an appropriate agent identity variable. Parallelism of agent sessions can also expressed in Casper.
Agent Sessions Description diagram: 	<pre> #FREE VARIABLES -- DEFINE AN AGENT VARIABLE AINITIATOR, ARESPONDER : AGENT #PROCESSES -- AGENT ROLES ARE MAPPED TO CSP PROCESSES -- PARAMETRISED WITH AGENT'S ID INITIATOR(AINITIATOR, <DATA VARIABLES>) RESPONDER(AINITIATOR, <DATA VARIABLES>) #ACTUAL VARIABLES -- CREATE AN INSTANCE FOR YOUR AGENT THE TERMINAL : AGENT #SYSTEM -- DECLARE POSSIBLE RUNS BY CREATING -- INSTANCES OF CSP PROCESSES INITIATOR(THE TERMINAL) RESPONDER(THE TERMINAL) </pre>

Figure 6. How to formalize the Agent's Sessions pattern

4. Distributed digital signatures case study

This case study was conducted within the R&D project sponsored by the Polish Scientific Council, Grant No. 6 T11 2003 C\0 6280 and led by Unizeto Ltd, Poland [20, 21]. The objective of the project was to develop and deploy a system supporting strong digital signatures in a distributed, public environment. The distributed architecture of the system and the new approach it takes to creating digital signatures resulted with the need to design a suit of secure cryptographic protocols. It was decided to analyze the protocols using formal techniques.

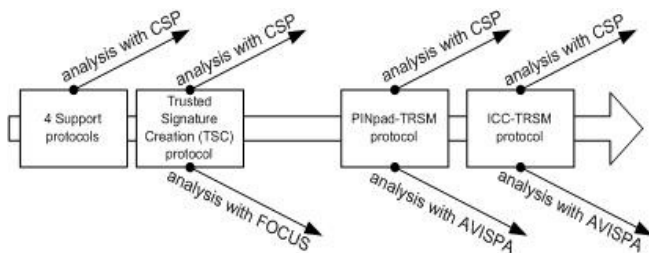


Figure 7. Protocols analyzed in the Unizeto Ltd. R&D project

The architecture of the developed system went through a significant change during the project, which directly influenced the designs of security protocols. The evolution of protocols and the formal analyses performed are shown in Figure 7.

The first suit of protocols consisted of five designs, the Trusted Signature Protocol (TSC) and four auxiliary ones [21]. TSC was responsible for establishing a secure channel between agents during the actual document signing. The protocol was modeled semi-formally and then analyzed formally using CSP/Casper and for additional verification with FOCUS. Later in the project, the signature creation transaction was divided into two smaller parts, the first one being an initial authentication of the pin-pad to the trusted computing module (PINpad-TRSM) and the second one authentication of the smartcard to the trusted module and producing a signature (ICC-TRSM). Both protocols have been analyzed using two formal methods: CSP/Casper and AVISPA.

The way the combination of our framework and the CSP/Casper formal method were applied to the Trusted Signature Protocol has already been reported in [22]. In the following section we will focus on explaining how the protocol was analyzed with FOCUS.

4.1 Trusted Signature Creation Protocol

The distributed system for digital signatures involves three types of agents: Application Provider, Service Provider and Signing Entity. The Application Provider (AP) represents a company or a government entity that is providing customers with an interactive web application which at some point requires submitting securely signed forms or documents. The Service Provider (SP) supplies the infrastructure and the service intended for creating digital signatures remotely. It is a trusted third party responsible for the overall security of the entire solution. The Signing Entity (SE) represents an independent party interested in using the application

reader and a PIN pad. The goal of the protocol is to establish a secure channel between the SE and SP, transmit the data to be signed to the smartcard held by SE and deliver the signed document first to SP and then to AP to finalize the transaction.

The UML model of the TSC protocol is presented in Figure 8. For clarity, we have removed the Application Provider from the model, as his involvement in the protocol is limited to receiving the signed data in the last message of the protocol. For the same reason we have omitted any <<assumption>> notes present in the full model. The notation used in the model is briefly explained in Figure 9.

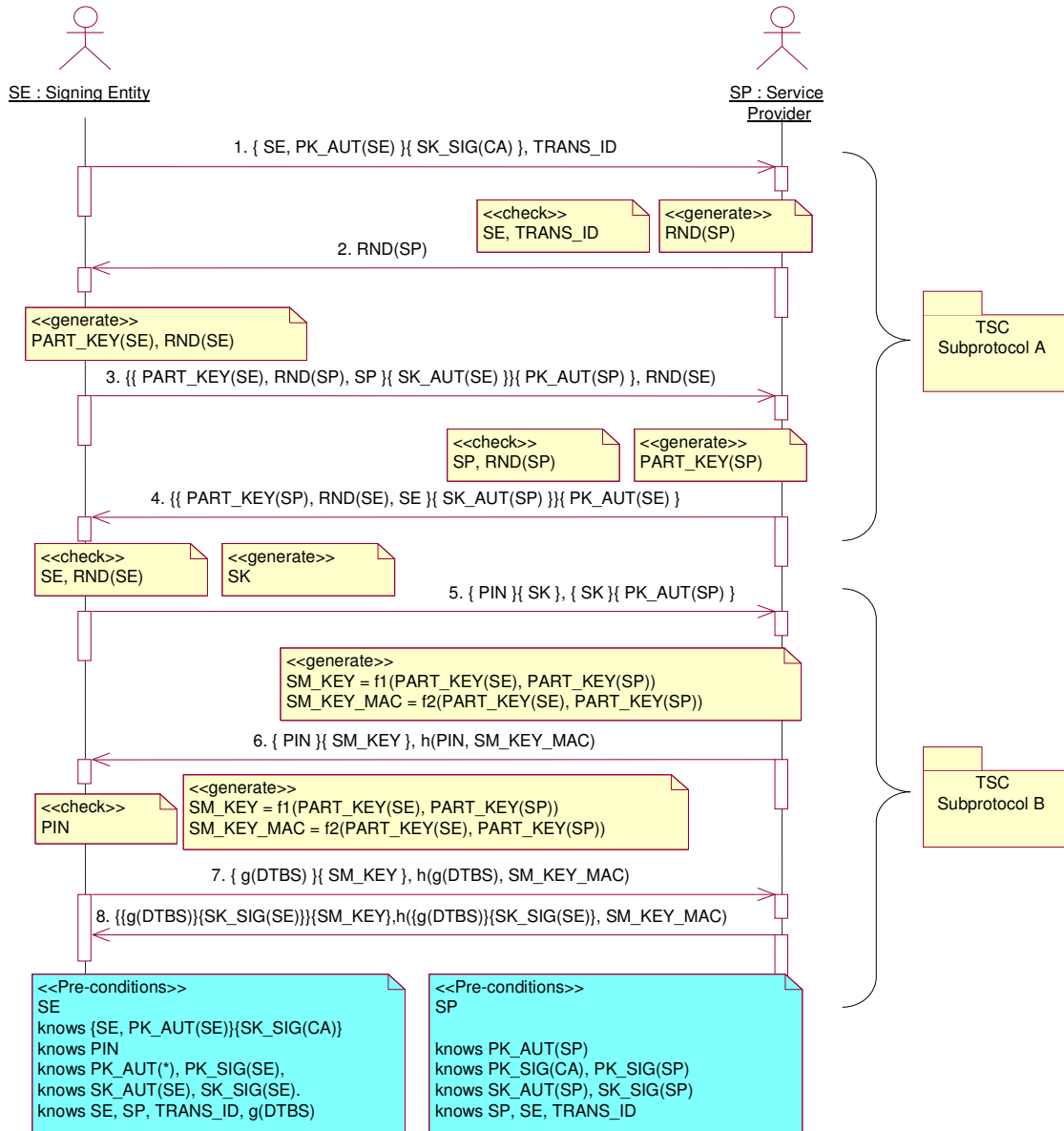


Figure 8. Trusted Signature Creation Protocol

offered by the AP. The service provider supplies the SE with a personal smartcard and the PIN number for accessing the private signature key stored on the card. The system is designed to allow the SE to sign electronic documents remotely on the Internet, providing he has access to a terminal equipped with a smartcard

4.2 Modeling the TSC protocol with FOCUS

The FOCUS formal method [9] together with its supporting tool, AutoFocus [23], were applied in the formal modeling process. The AutoFocus tool was originally intended for developing reliable embedded systems. It was adapted to cryptographic

Symbol	Definition
M_i (for $i \in N$)	atomic data or a data structure
$M_1, M_2, M_3 \dots$	concatenation of fields $M_1, M_2, M_3 \dots$
$\{M\}\{K\}$	message M encrypted or signed using K
$PK_AUT(A), SK_AUT(A)$	A 's public and private key for encryption
$PK_SIG(A), SK_SIG(A)$	A 's public and private key for signatures
$\llchecked\gg M_1, M_2, M_3 \dots$	an agent checks values of $M_1, M_2, M_3 \dots$ after receipt
$\llgenerated\gg M_1, M_2, M_3 \dots$	an agent generates fresh values of $M_1, M_2, M_3 \dots$
$TRANS_ID$	a transaction (session) identification number

Symbol	Definition
K	key used for encryption or signatures
A	agent or a Certification Authority
SK, SM_KEY	a shared session key
PIN	SE 's private identification number
$M = f(M_1, M_2, M_3 \dots)$	M is generated using values $M_1, M_2, M_3 \dots$
$h(M_1, M_2, M_3 \dots), g(M_1, M_2, M_3 \dots)$	one-way hash functions
$DTBS$	data to be signed by SE
$RND(A), PART_KEY(A)$	random numbers generated by A

Figure 9. Notation used in the model of the TSC protocol

protocol analysis because FOCUS allows modeling of all relevant cryptographic operations and functions [24]. However, the flexibility of the formal language and its universal nature result in a relatively high degree of model complexity.

A security protocol specification in FOCUS can be documented using a number of diagrams. Agents and communication channels that connect them are modeled with System Structure Diagrams (SSD). Internal behavior of agents is represented using State Transition Diagrams (STD). Sequence diagrams, which are used to represent how a protocol session is run, are shown as extended Event Traces Diagrams (EET). Finally, types of messages and variables as well as functions are declared as Data Type Definition (DTD) specifications.

The biggest challenge which we encountered in the process of the protocol analysis, was the complexity of the resulting protocol specification. Experiments with formalization and verification of the entire protocol model failed. This led to a decision that decomposition of the model is necessary. The TSC protocol was divided into two parts, as shown in Figure 8.

The first part – TSC Subprotocol A - consists of the first four messages. The objective of Subprotocol A is to have the Signing Entity (SE) and Service Provider (SP) successfully establish values of secret keys SM_KEY and SM_KEY_MAC . The second part – TSC Subprotocol B - consists of the last five messages (four, if you do not count communication with the Application Provider at the very end of the session). The objective of Subprotocol B is to provide the Application Provider (AP) and the SP with a properly signed hash of the document to be signed: $g(DTBS)$. This goal is achieved with the help of a secret pair of keys: SM_KEY and SM_KEY_MAC established as a result of Subprotocol A. Separate formal models were created for both subprotocols. Then it was proved in a classical, deductive way that if both subprotocols are secure then the whole scheme is secure as well.

Specifications for both subprotocols included three active parties: two legitimate agents SE and SP and the intruder. AP was not

explicitly included in the model. The role of AP was taken into account by redirecting the last message of the protocol originally intended for him to SE. This modification allowed us to limit the complexity of the resulting specification. At the same time it was proven that the transformation was secure, which means it did not influence the results of model verification. The specification of TSC agents is shown in Figure 10.



Figure 10. Simplified formal model of the communication parties of the protocol

For both subprotocols the intruder is modeled by two subcomponents: Fake-Store and Overhear as shown in Figure 11. The first component is responsible for storing messages, performing analyses, and creating faked messages based on the acquired knowledge. The second component models the control intruder has over the network, his ability to capture messages or introduce messages from Fake-Store into the network. Such a model is consistent with the mathematical definition of the Dolev-Yao intruder.

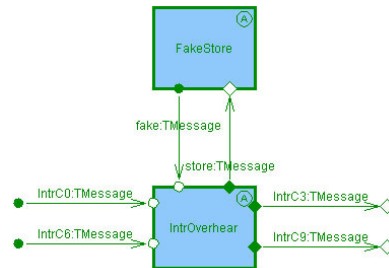


Figure 11. Formal model of the intruder

The next step was to create a state diagram for each FOCUS component (agents and the intruder). Our model permitted only a single protocol session between SE and SP and one session

between SP and the intruder acting as a dishonest user. However it was proved that this assumption would not influence the results of model verification.

4.3 Verification and results

Properties of our formal models of the TSC protocol were analyzed using a model checker. For Subprotocol A, formal showed that both halves of the two secret keys: SM_KEY and SM_KEY_MAC generated by the SE and SP are exchanged and authenticated correctly and their secrecy is not compromised.

The objective of formal verification of Subprotocol B was to show that secret values of SE's PIN and the session key SK are not intercepted and that every message accepted by the AP agent contains a legitimate signed document hash issued by SE. However, a potential flaw was discovered in the protocol. The protocol has an anomaly which arguably allows the intruder to intercept user's PIN number.

The attack can be performed in the following way:

- The intruder intercepts and stores message five from the session between SE and SP – $\{\text{PIN}\}\{\text{SK}\},\{\text{SK}\}\{\text{PK_AUT}(\text{SP})\}$.
- He repeats the captured message during his own session as a legitimate user.
- As a reply, the intruder receives a message $\{\text{PIN}\}\{\text{SM_KEY_A}\}$, where SM_KEY_A is a secret key known only by SP and the intruder.
- From this message PIN is retrieved as the intruder knows SM_KEY_A.

The protocol has an anomaly because one of rules of robust protocol design was broken i.e. every message should be cryptographically linked with one or more other messages in the same session. Message 5 is open to a simple replay attack. An example solution to this problem could be to include TRANS_ID in the fifth message encrypting it with the session key SK alongside the PIN.

It is worth mentioning that analysis of the same protocol conducted using CSP/Casper and reported in [22] discovered the same anomaly. The issue was corrected by the protocol designers. No further anomalies were discovered in any of the other protocols designed for the distributed digital signatures system.

5. PBK Case Study

The subject of this case study was analysis of the Purpose-Built Keys (PBK) protocol. Our objective was to take a protocol that we know has at least one vulnerability, analyze it using our framework and at least two different formal methods and compare the results. An additional requirement was that original design documentation must be available for the protocol, and not just an abstract model. Only in such a scenario we would be able to test all the features of our integrated framework.

The have selected the PBK protocol, since there is a standard Internet Draft specification document available for it [25] and one of the past versions (revision 6) of the protocol had been found defective by the AVISPA project team [26].

5.1 Analysis

For this case study we used CSP/Casper and AVISPA. Of course, in case of the latter method our formal model was created completely independently from the one prepared by the AVISPA project team in their experiments. Specifications for both AVISPA and CSP/Casper were created based on a shared UML model in accordance to the analytical procedure and formalization rules included in the framework. One of the UML diagrams modeling the protocol, in this case showing the protocol dynamics is presented in Figure 12.

The purpose of Purpose-Build Keys protocol is to guarantee authentication of origin of the Request sent from the Initiator to the Responder. For each protocol session, the Initiator generates a fresh public/secret key pair and uses the secret key to sign the Request sent in message two. The public part of the key pair is transmitted to the Responder in the first message. It is crucial however that this first transmission reached the responder reliably (message integrity and authentication of origin must be ensured), there is no other way to bind the key to the Initiator's identity. This requirement is explicitly stated in protocol specification and resulted in an assumption **A_secure_initialization** shown on our model as well as a special stereotype <<secure channel>> for the first protocol message. After the Request is sent there is an additional Challenge-Response exchange to confirm that the originating agent is indeed the holder of the private key sent during the initialization.

5.2 Results

This protocol has an anomaly which arguably may become a vulnerability. The problem comes from messages three and four. The Responder is expected to sign the Challenge which is chosen arbitrarily by the Initiator. This is a violation of the "don't sign random strings of bits" rule of protocol design. In fact the intruder can use an unsuspecting honest agent as an oracle by providing him a faked Request to sign as a seemingly "random" challenge. Then he can use the signed faked Request during another session thus violating authentication of origin.

We attempted to rediscover this attack using CSP/Caper and AVISPA. The first result was that this protocol can not be successfully analyzed using CSP/Casper. This fact became apparent during formalization phase, when it became clear that authentication of origin requirement can not be expressed in the language of Casper – it only supports entity authentication. Entity authentication is too restrictive for this protocol and would result in finding "dummy" attacks.

AVISPA has somewhat greater flexibility in terms of what security goals can be verified and we managed to rediscover the attack using this formal method. However, the only reason this attack is successful is the type flaw between messages two and four (specifically between $\{\text{Request}\}\{\text{PrivKey}\}$ and $\{\text{Challenge}\}\{\text{PrivKey}\}$). The original Internet Draft specification states that "If replay protection is necessary, a nonce value (...) or timestamp may be included with the operation request." [25]. That means the attack can be successful only if we assume that the Request is sent without the nonce and model the protocol accordingly. We have included the **A_request_is_signed_without_nonce** assumption in the analysis report and also put it in the model shown in Figure 12. This additional information allows to trace back the reason for the

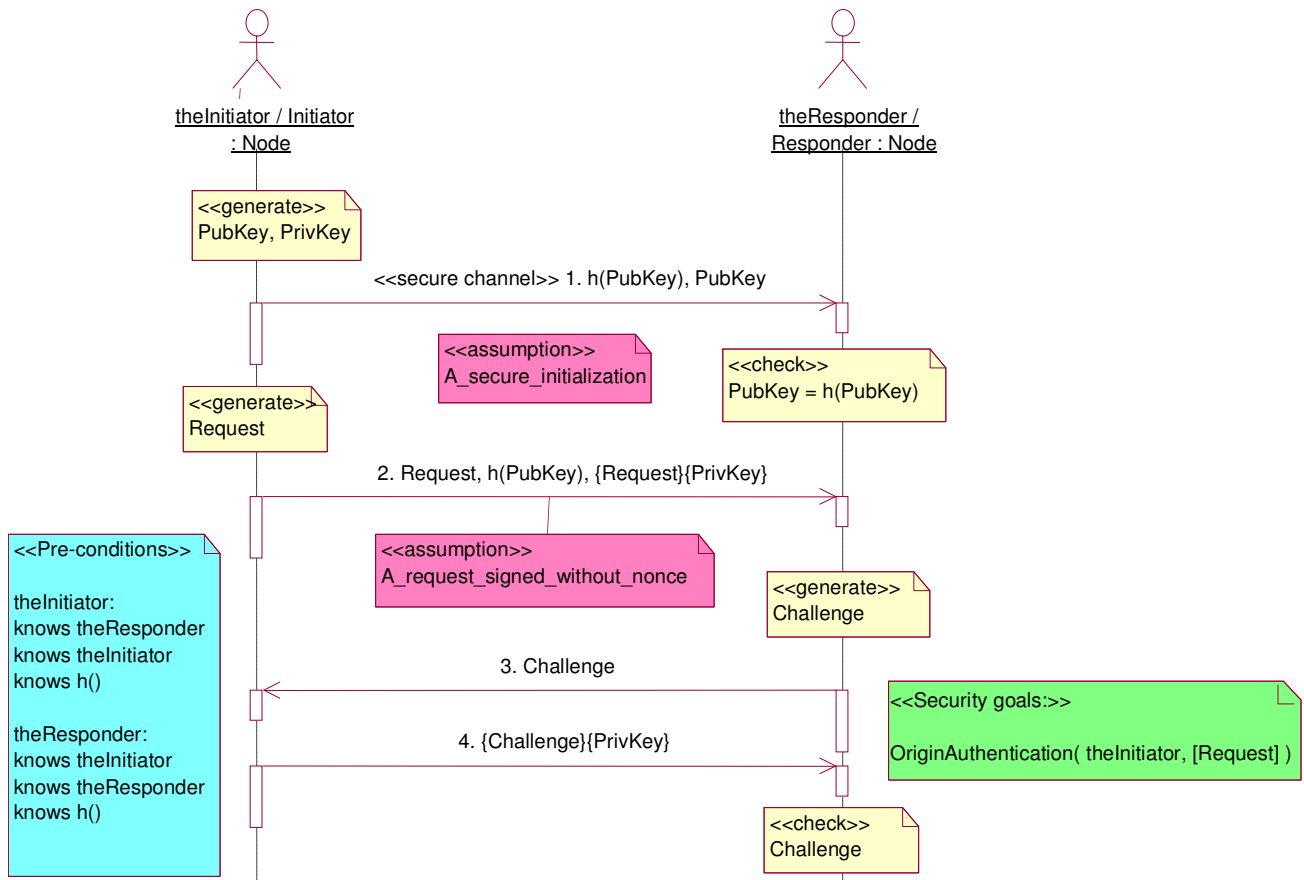


Figure 12. Protocol dynamics of the PBK

attack to a modeling decision that was made earlier in the process, even before formalization. This assumption was not given explicitly by the team at AVISPA project that analyzed this protocol.

To summarize: we managed to rediscover the attack through independent modeling and analysis. The problems we came across with CSP/Casper show that diversity of formal methods available at hand in any given project may become indispensable. A simple example of an important assumption present in the PBK protocol model demonstrates the added value offered by assumption traceability.

6. Conclusions

The paper introduced a framework that integrates object and formal modeling in order to support the process of analysis of security protocols. The framework uses extended (by means of stereotyping and profiling) UML for object modeling and several formal model checking methods. The transition between UML models and formal specifications is facilitated by using a set of UML modeling patterns and formalization rules. The process of modeling and analysis is well supported by existing tools, such as UML CASE tools and others depending on the formal methods applied: AVISPA, Casper compiler and FDR model checker or AutoFocus.

The framework encompasses a formalized analytical procedure which guides a user through a sequence of well defined steps. The process is supported by a set of documentation templates and

patterns with a particular attention on identifying and documenting the security objectives and all the assumptions that condition the results of the analyses. We have found that such explicit and comprehensive documentation of all identified assumptions (including their motivation, rationale and possible consequences) together with easy to understand UML models was very effective in two areas. The first was establishing communication between the analysts and the protocol designers, and the second was tracing back all the analytical decisions made throughout the procedure.

The biggest challenge in applying the framework is still the size and complexity of formal models for industry security protocols, such as the one analyzed within the distributed digital signatures case study. Our experience in analyzing the protocol with FOCUS and CSP/Casper (results reported in [22]) is that it involves specialized techniques for managing complexity, such as proving smaller properties first and applying the outcomes to simplify the main model. This requires extensive experience with protocol analysis as well as expert knowledge of the tools. Our framework is not a “push-button” technology. However, the more complex the analysis process becomes, the greater value there is to be gained by having a system for controlling the focus and scope of formal modeling and for documenting and communicating the results.

In the nearest future we plan for further development of the framework and for running more case studies, perhaps with additional formal techniques. We are also developing a software tool which will support the workflow of our analytical procedure

and assist with managing all the artifacts created during the analysis.

7. REFERENCES

- [1] C. Fidge, "A Survey of Verification Techniques for Security Protocols Technical Report 01-22", Software Verification Research Centre, School of Information Technology, The University of Queensland, 2001.
- [2] R. Anderson, "Security Engineering", Wiley, ISBN: 0-471-38922-6, 2001.
- [3] C. A. Meadows, "*Formal Verification of Cryptographic Protocols: A Survey*", ASIACRYPT: Advances in Cryptology, 1995.
- [4] M. Burrows, M. Abadi, R. Needham, "A logic of authentication. Technical Report TR 39", Digital Equipment Corporation, February 1989.
- [5] L. Paulson, "The inductive approach to verifying cryptographic protocols", University of Cambridge Computer Laboratory, December 1998.
- [6] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, L. Vigneron, "The Avispa Tool for the automated validation of internet security protocols and applications", Computer Aided Verification, LNCS 3576, 2005.
- [7] F. J. Thayer, J.C. Herzog, J.D. Guttman, "Strand spaces: Why is a security protocol correct?", Proceedings of 1998 IEEE Symposium on Security and Privacy, 1998.
- [8] A.W. Roscoe, "The Theory and Practice of Concurrency", Prentice-Hall, International Series in Computer Science, ISBN 0-13-674409-5, 1998.
- [9] B. Broy, F. Dederichs, M. Fuchs, T.F. Gritzner, R. Weber, "The Design of Distributed Systems – An Introduction to FOCUS", SFB-Report 342/2-2/92 A, Technical University of Munich, 1993.
- [10] M. Abadi, A. D. Gordon, "A calculus for cryptographic protocols: The Spi Calculus", In Proceedings of the 4th ACM Conference on Computer and Communications Security, ACM Press, 1997.
- [11] L. Vigneron, "Specification Languages for Internet Security Protocols", Workshop on Automated Validation of Internet Security Protocols and Applications, 2004.
- [12] J. M. Bruel, "Integrating Formal and Informal Specification Techniques. Why? How?", Second IEEE Workshop on Industrial Strength Formal Specification Techniques, 1998.
- [13] J. M. Bruel, R. B. France, "Transforming UML models to Formal Specifications", International Conference on the Unified Modelling Language (UML): Beyond the Notation, 1998.
- [14] E. Boiten, M. Bujorianu, "Exploring UML Refinement through Unification", Critical Systems Development with UML - Proceedings of the UML'03 workshop, number TUM-I0323, pages 47-62, Technische Universitat Munchen, September 2003.
- [15] J. Jurjens, "Secure Systems Development with UML", Springer, ISBN: 3-540-00701-6, 2004.
- [16] CASENET, European Union 5th Framework Program project, IST-2001-32446
- [17] J. Rumbaugh, I. Jacobson, G. Booch, "The Unified Modelling Language Reference Manual (2nd Edition)", Addison-Wesley, 2005.
- [18] S. Johnston, "Rational UML Profile for business modeling", IBM Corp. whitepaper, 2004.
- [19] B. Selic, "Unified Modeling Language version 2.0", IBM Corp. whitepaper, 2005.
- [20] W. Chocianowicz, J. Pejas, A. Rucinski, "The Proposal of Protocol for Electronic Signature Creation in Public Environment", in Enhanced Methods in Computer Security, Biometric and Artificial Intelligence Systems, Kluwer Academic Publishers, ISBN 1-4020-7776-9, 2005.
- [21] W. Chocianowicz, W. Mackow, A. Skrobek, P. Sukiennik, J. Pejas, Project No. 6 T11 2003 C/0 6280 – Technical Report 7: "Design and implementation of an universal module for reliable presentation of a document to be signed or verified", Szczecin University of Technology, 2004.
- [22] M. Olszewski, "A Model-based Approach to Analysis of Security Protocols – A Case Study", Proceedings of the Technologies for Homeland Security and Safety Conference, Poland, 2005.
- [23] J. Jurjens, G. Wimmel, "Formally Testing Fail-Safety of Electronic Purse Protocols", 2001.
- [24] J. Grunbauer, H. Hollmann, J. Jurjens, G. Wimmel, "Modelling and Verification of Layered Security Protocols: A Bank Application", Computer Safety, Reliability, and Security, 22nd International Conference SAFECOMP, 2003.
- [25] S. Bradner, A. Mankin, J. Schiller, "A Framework for Purpose-Built Keys (PBK)", IETF Internet-Draft: draft-bradner-pbk-frame-06.txt, 2003.
- [26] The AVISPA Library of protocols, <http://www.avispa-project.org/>, AVISPA project: IST-2001-39252