Constructing Security Protocol Specifications for Web Services

Genge Bela, Haller Piroska, and Ovidiu Ratoi

"Petru Maior" University of Targu Mures, Electrical Engineering Department
Nicolae Iorga St., No. 1, Mures, RO-200440, Romania
{bgenge,phaller,oratoi}@engineering.upm.ro

Summary. In order to integrate new security protocols, existing systems must be modified accordingly, which often means interrupting system activity. We propose a solution to this problem by developing an ontology model which provides semantic to security protocol operations. The proposed model is based on a formal specification model and is integrated in existing Web service description technologies.

1 Introduction

Security protocols are widely used today to provide secure communication over insecure environments. By examining the literature we come upon various security protocols designed to provide solutions to specific problems ([10]). With this large amount of protocols to chose from, distributed heterogenous systems must be prepared to handle multiple security protocols.

Existing technologies, such as the Security Assertions Markup Language ([12]) (i.e. SAML), WS-Trust ([15]) or WS-Federation ([14]) provide a unifying solution for the authentication and authorization issues through the use of predefined protocols. By implementing these protocols, Web services authenticate users and provide authorized access to resources. However, despite the fact that existing solutions provide a way to implement security claims, these approaches are rather static. This means that in case of new security protocols, services supporting the mentioned security technologies must be reprogrammed.

In this paper, we propose a more flexible solution to this problem by developing an ontology model aiming at the automatic discovery and execution of security protocols. An ontology is a "formal, explicit specification of a shared conceptualization" ([1]), consisting of concepts, relations and restrictions. Ontologies are part of the semantic Web technology, which associates semantic descriptions to Web services.

In order to construct the proposed ontology model, we first create an enriched formal specification model for security protocols. In addition to the information provided by existing formal models, such as the SPI calculus ([2]), the strand space model ([3]) or the operational semantics ([4]), we also include explicit processing operations. These operations are then translated to semantic concepts and properties in the proposed ontology model.

C. Badica et al. (Eds.): Intel. Distributed Comput., Systems & Appl., SCI 162, pp. 245–250, 2008.
springerlink.com

The paper is structured as follows. In section 2 we construct a formal security protocol specification model. Based on this, in section 3 we describe the proposed ontology model and an example implementation. In section 4 we connect our work to others. We end with a conclusion and future work in section 5.

2 Security Protocol Specifications

Existing security protocol specifications limit themselves to the representation of operations and message components that are vital to the goal of these protocols: exchanging messages in a secure manner. One of the most simplest form of specification is the *informal* specification. For example, let us consider Lowe's modified version of the BAN concrete Andrew Secure RPC ([5]):

$$
\begin{aligned}
&A \rightarrow \mathrm{B}: A, N_a \\
&B \rightarrow \mathrm{A}: \{N_a, K, B\}_{K_{AB}} \\
&A \rightarrow \mathrm{B}: \{N_a\}_K \\
&B \rightarrow \mathrm{A}: N_b
\end{aligned}
$$

By running the protocol, two participants, A and B, establish a fresh session key K. The random number N_a ensures freshness of the newly generated key, while N_b is sent by participant B to be used in future sessions. Curly brackets denote symmetrical key encryption. Throughout this paper we use the term "nonce", which is a well-known term in the literature, to denote random numbers.

Participants running security protocols usually exchange message components belonging to well-defined categories. We model these categories using the following sets: R, denoting the set of participant names; N, denoting the set of nonces and K, denoting the set of cryptographic keys.

The message components exchanged by participants are called *terms*. *Terms* may contain other terms, encrypted or not. Encryption is modeled using *function names*. The definition of *function names* and *terms* is the following:

$$
\begin{aligned}
\mathit{FuncName} \ ::= & \ \mathrm{sk} \quad (\mathit{secret\ key}) \\
| & \ \mathrm{pk} \quad (\mathit{public\ key}) \\
| & \ \mathrm{h} \quad (\mathit{hash\ or\ keyed\ hash})
\end{aligned}
\qquad
\begin{aligned}
\mathcal{T} ::= . \ | \ & \mathsf{R} \mid \mathsf{N} \mid \mathsf{K} \mid (\mathcal{T}, \mathcal{T}) \\
| \ & \{\mathcal{T}\}_{\mathit{FuncName}(\mathcal{T})}
\end{aligned}
$$

Terms that have been encrypted with one key can only be decrypted by using either the same key (when dealing with symmetric encryption) or the inverse key (when dealing with asymmetric encryption). To determine the corresponding inverse key, we use the $\_^{-1} : \mathsf{K} \rightarrow \mathsf{K}$ function.

As opposed to regular specifications where the user decides on the meaning of each component, for our goal to be achievable, we need to include additional information in the specification so that protocols can be executed without any user intervention.

We use the term "protocol header" to denote a set of sections needed for the interpretation of the information that follows. The header we propose consists of three sections: *types*, *precondition* and *effect*. The predicates defined for each section are given in table 1. Using the defined terms, we now define several functions to operate on them,

Table 1. Predicate definitions used to construct the protocol header

Section	Predicate	Definition	Description
types	*part*	R^*	Participant list
	nonce	N^*	Nonce list
	key	K^*	Key list
	term	$\mathcal{T}^*$	Term list
precondition	*shared_key*	$\mathsf{R} \times \mathsf{R} \times \mathsf{K}$	Shared key between two participants
	init_part	R	Initializing participant
	resp_part	R	Respondent participant
effect	*key_exchange*	$\mathcal{T}^*$	Key exchange protocol
	authentication	$\mathcal{T}^*$	Authentication protocol

Table 2. Function definitions used to construct the protocol body

Function	Definition	Example Usage : Result
gennonce	$\mathsf{R} \rightarrow \mathsf{N}$	$gennonce(A) : N_a$
genkey	$\mathsf{R} \rightarrow \mathsf{K}$	$genkey(A) : K$
encrypt	$\mathsf{R} \times \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$	$encrypt(A, (A, N_a), K_{ab}) : \{A, N_a\}_{sk(K_{ab})}$
decrypt	$\mathsf{R} \times \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$	$decrypt(A, \{A, N_a\}_{sk(K_{ab})}, K_{ab}^{-1}) : (A, N_a)$

resulting the "protocol body". These functions are used to provide a detailed description of atomic operations specific to term construction and extraction. Sending and receiving operations are handled by $send : \mathsf{R} \times \mathsf{R} \times \mathcal{T} \rightarrow \mathcal{T}$ and $recv : \mathsf{R} \times \mathsf{R} \times \mathcal{T} \rightarrow \mathcal{T}$ functions.

The list of proposed functions is given in table 2, which can be extended with other functions if needed.

3 Ontology Model and Semantic Annotations

Based on the formal protocol construction from the previous section we have developed an ontology model that serves as a common data model for describing semantic operations corresponding to security protocol executions. The core ontology (figure 1) defines a security protocol constructed from four domains: *Cryptographic specifications*, *Communication*, *Term types* and *Knowledge*. Interrupted lines denote ontology import, empty arrowed lines denote sub-concept association and filled arrowed lines denote functional relations (from domain to range) between concepts.

The proposed ontology has been developed in the *Protégé* ontology editor ([9]). It provides semantic to protocol operations such as generating new terms (i.e. key or nonce), verifying received terms, sending and receiving terms.

The knowledge concept plays a key role in the automatic execution process. It's purpose is to model the stored state of the protocol between exchanged messages. For example, after generating a new nonce, this is stored in the knowledge of the executing

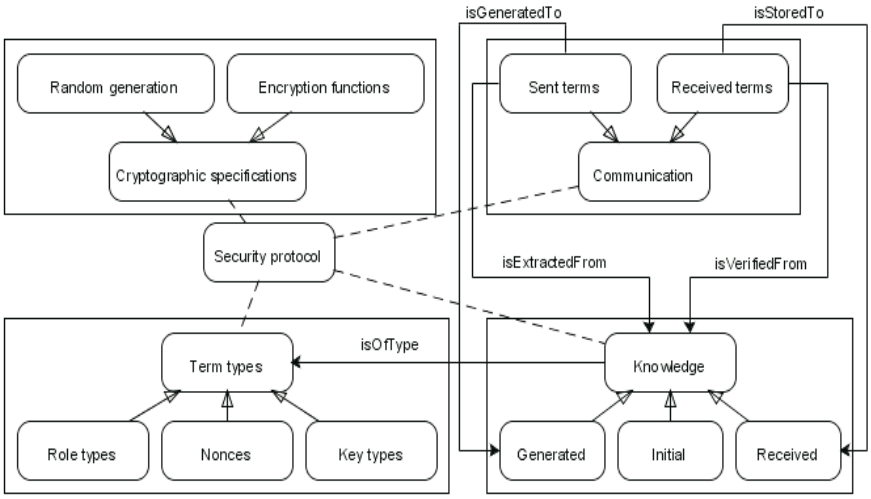

Fig. 1. Core ontology for describing security protocols

participant. When a term is received containing the same nonce, the participant must verify it's validity by comparing the stored value with the received one.

The ontology model provides semantic for the purpose of each sent and received term. However, it does not provide description of the mechanisms that would allow terms to be exchanged by parties. It also does not provide a specification of the preconditions, effects, cryptographic details or exchanged message sequences.

In our implementations, we used WSDL with Semantic annotations ([11]) (i.e. WSDL-S) to handle the aspects that are not addressed by the ontology model. The WSDL standard defines the *portType* section to denote a collection of messages. We annotate messages defined by *portType* using the *wssem:modelReference* extension attribute. Part of an example annotated XML schema representation of an encrypted message is the following:

```
<complexType name="encType">
    <sequence>
        <element name="role" type="tns:roleType"
         wssem:modelReference="http://www.owl-ontologies.com/sec.owl#RoleA">
        </element>
        <element name="nonce" type="tns:nonceType"
         wssem:modelReference="http://www.owl-ontologies.com/sec.owl#NonceA>
        </element>
        <element name="keyRef" type="tns:sharedKeyType"
         wssem:modelReference="http://www.owl-ontologies.com/sec.owl#KeyAB>
        </element>
        <element name="encAlg" type="string"
         wssem:modelReference="http://www.owl-ontologies.com/sec.owl#EncAES>
        </element>
    </sequence>
</complexType>
```

Effects and preconditions are added using the already existing *wssem:precondition* and *wssem:effect* elements. Part of the WSDL-S defining a security protocol which

requires a shared key between users and as an effect produces a session key is the following:

```
<wsdl:portType name="EncComm">
  <wssem:precondition name="SharedKey"
   wssem:modelReference="http://www.owl-ontologies.com/sec.owl#SharedKeyAB"/>
  <wssem:precondition name="SessionKey"
   wssem:modelReference="http://www.owl-ontologies.com/sec.owl#SessKeyAB"/>
</wsdl:portType>
```

4 Related Work

There are several proposals and already established standards in the web service community dealing with security aspects. We will briefly describe three approaches that have the most in common with our proposal: SAML (Security Assertion Markup Language), WS-Federation (Web Service-Federation) and WS-Security (Web Service-Security).

The Security Assertion Markup Language is an "XML framework for exchanging authentication and authorization information" ([12]) between entities. This is achieved by the use of assertions, based on XML constructions, that denote authentication and authorization sequences achieved by entities. Based on these assertions, service providers can decide whether clients are authorized or not to access the requested services. SAML also provides a set of XML constructions called "profiles" to describe the required message exchange for transferring assertions. However, these are predefined messages that must be implemented by all services and protocol participants. Our proposal includes a semantic description that allows executing security protocols containing message structures that are not predefined.

A similar proposal to the SAML framework is WS-Federation ([14]). As mentioned by the authors of WS-Federation, the goals achieved are mainly the same as in the case of SAML. Major differences relate to the fact that it extends the WS-Trust model ([15]) and it provides a set of composable protocols. These components that differentiate it from the SAML framework, however, do not compete with our proposal.

The WS-Security ([13]) proposes a standard set of SOAP extensions to implement message integrity and confidentiality. It describes how to encode binary security components such as keys, random numbers or X.509 tokens. The WS-Services specification is thus a transport layer for the actual execution of protocols and not the description of the involved messages. The WS-Services could thus be used in conjunction with our proposal to encode binary data included in protocol messages.

5 Conclusions and Future Research

In this paper we presented an ontology model which provides semantic to security protocol execution operations. The ontology model is based on a formal specification model that ensures a detailed description through the use of a protocol header and a protocol body. Based on these descriptions, client applications can execute new security protocols based only on the proposed description.

The major advances in the field of protocol composition ([7, 8]) provide the means to create new protocols from existing security protocols. The analysis process of the composed protocols has been reduced to a syntactical analysis ([6]) which could be used to create protocols in real time. As future research we intend to combine the specification and ontology model proposed in this paper with composition operators in order to create protocols based on multiple specifications.

References

1. Studer, R., Benjamins, V., Fensel, D.: Knowledge Engineering: Principles and Methods. In: Data and Knowledge Engineering, pp. 161–197 (1998)
2. Abadi, M., Gordon, A.D.: A Calculus for Cryptographic Protocols: the spicalculus. In: 4th ACM Conference on Computer and Communications Security, pp. 36–47 (1997)
3. Fabrega, F.J.T., Herzog, J.C., Guttman, J.D.: Strand Spaces: Why is a security protocol correct? In: Proc. Of the 1998 Symposium on Security and Privacy, pp. 66–77 (1998)
4. Cremers, C., Maw, S.: Operational semantics of security protocols. In: Leue, S., Systa, T. (eds.) Scenarios: Models, Transformations and Tools. LNCS, vol. 3466, pp. 66–89. Springer, Heidelberg (2005)
5. Lowe, G.: Some new attacks upon security protocols. In: Proc. of the 8th Computer Security Foundations Workshop (1996)
6. Genge, B., Ignat, I.: Verifying the Independence of Security Protocols. In: Proc. of the 3rd International Conference on Intelligent Computer Communication and Processing, Romania, pp. 155–163 (2007)
7. Datta, A., Derek, A., Mitchell, J.C., Roy, A.: Protocol Composition Logic. In: Electronic Notes in Theoretical Computer Science, pp. 311–358 (2007)
8. Hyun-Jin, C.: Security protocol design by composition. Technical report Nr. 657, UCAM-CL-TR-657, Cambridge University, UK (2006)
9. Noy, N.F., Crubezy, M., et al.: Protege-2000: An Open-Source Ontology-Development and Knowledge-Acquisition Environment. In: AMIA Annual Symposium Proceedings (2003)
10. Security Protocol Open Repository (2008), http://www.lsv.ens-cachan.fr/spore/
11. World Wide Web Consortium. Web Service Semantics WSDL-S Recommendation (November 2005), http://www.w3.org/TR/wsdl/
12. Organization for the Advancement of Structured Information Standards. SAML V2.0 OASIS Standard Specification (November 2007), http://saml.xml.org/
13. Organization for the Advancement of Structured Information Standards. OASIS Web Services Security (WSS) TC (November 2006), www.oasis-open.org/committees/wss/
14. IBM. Web Services Federation Language Specification (December 2006), http://www.ibm.com/developerworks/library/specification/ws-fed/
15. Organization for the Advancement of Structured Information Standards. WS-Trust v1.3 OASIS Standard (March 2007), http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html/