# Towards Automated Secure Web Service Execution

Béla Genge and Piroska Haller

"Petru Maior" University of Târgu Mureş, Department of Electrical Engineering,
N. Iorga Str., No. 1, (540088) Târgu Mureş, ROMANIA,
{bgenge,phaller}@engineering.upm.ro
http://www.upm.ro

**Abstract.** Existing solutions for authentication and authorization in Web services make use of technologies such as SAML or WS-Security. These provide a static solution by using a set of predefined protocols. We propose a semantic security protocol model from which security protocol specifications are generated and automatically executed by participants. The proposed model consists of a sequential component, implemented as a WSDL-S specification, and an ontology component, implemented as an OWL specification. The correctness of the proposed model is ensured by using a set of rules and algorithms for generating it based on a protocol model given by the user. We validate our approach by generating and implementing several specifications for existing protocols such as ISO9798 or Kerberos protocols.

**Key words:** Security protocols, automated execution, ontology, Web services

## 1 Introduction

Security protocols are "communication protocols dedicated to achieving security goals" (C.J.F. Cremers and S. Mauw) [1] such as confidentiality, integrity or availability.

Existing technologies, such as the Security Assertions Markup Language [16] (i.e. SAML) or WS-Security [17] 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 these approaches are rather static, meaning that in case of new security protocols, services must be reprogrammed.

In this paper we propose a semantic security protocol model (SSPM) for generating security protocol specifications that can be automatically executed by participants. The SSPM has two components: a sequential model and an ontology model. The first component is implemented as a WSDL-S [4] specification while the second component is implemented as an OWL [12] specification. The role of the WSDL-S implementation is to describe the message sequences and directions that must be executed by protocol participants.

The role of the OWL implementation is to provide semantic information such as the construction, processing and implementation of cryptographic operations (e.g. encryption algorithm, encryption mode, key).

The SSPM is constructed from a given SPM and it must maintain the protocol's security properties. For this we propose several generating rules and algorithms that provide a mapping for each component from SPM to SSPM. The correctness of the proposed rules and algorithms results from the one-to-one mapping of each component and from the correctness of SPM.

The paper is structured as follows. In section 2 we present the basic protocol model. In section 3 we present the semantic model with generating rules and algorithms. Example usage and experimental results are presented in section 4. We relate our work to others in section 5. We end with a conclusion and future work in section 6.

## 2 Protocol Model

Protocol participants communicate by exchanging *terms* constructed from elements belonging to the following basic sets: $\mathsf{P}$, denoting the set of role names; $\mathsf{N}$, denoting the set of random numbers or *nonces* (i.e. "number once used"); $\mathsf{K}$, denoting the set of cryptographic keys; $\mathsf{C}$, denoting the set of certificates and $\mathsf{M}$, denoting the set of user-defined message components.

In order for the protocol model to capture the message component types found in security protocol implementations [16], [17] we specialize the basic sets with the following subsets:

- $\mathsf{P}_{DN} \subseteq \mathsf{P}$, denoting the set of distinguished names; $\mathsf{P}_{UD} \subseteq \mathsf{P}$, denoting the set of user-domain names; $\mathsf{P}_{IP} \subseteq \mathsf{P}$, denoting the set of user-ip names; $\mathsf{P}_U = \{\mathsf{P} \setminus \{\mathsf{P}_{DN} \cup \mathsf{P}_{UD} \cup \mathsf{P}_{IP}\}\}$, denoting the set of names that do not belong to the previous subsets;
- $\mathsf{N}_T$, denoting the set of timestamps; $\mathsf{N}_{DH}$, denoting the set of random numbers specific to the Diffie-Hellman key exchange; $\mathsf{N}_A = \{\mathsf{N} \setminus \{\mathsf{N}_{DH} \cup \mathsf{N}_T\}\}$, denoting the set of random numbers;
- $\mathsf{K}_S \subseteq \mathsf{K}$, denoting the set of symmetric keys; $\mathsf{K}_{DH} \subseteq \mathsf{K}$, denoting the set of keys generated from a Diffie-Hellman key exchange; $\mathsf{K}_{PUB} \subseteq \mathsf{K}$, denoting the set of public keys; $\mathsf{K}_{PRV} \subseteq \mathsf{K}$, denoting the set of private keys;

To denote the encryption type used to create cryptographic terms, we define the following *function names*:

$$\begin{aligned} FuncName ::= {} & sk && (symmetric\, function) \\ \mid {} & pk && (asymmetric\, function) \\ \mid {} & h && (hash\, function) \\ \mid {} & hmac && (keyed\, hash\, function) \end{aligned}$$

The encryption and decryption process makes use of cryptographic keys. Decrypting an encrypted term is only possible if participants are in the possession of the decryption key pair. In case of symmetric cryptography, the decryption key is the same as the encryption key. In case of asymmetric cryptography, there is a public-private key pair. Determining the corresponding key pair is done using the function $\_^{-1} : \mathsf{K} \rightarrow \mathsf{K}$.

The above-defined basic sets and function names are used in the definition of *terms*, where we also introduce constructors for pairing and encryption:

$$\mathsf{T} ::= . \mid \mathsf{P} \mid \mathsf{N} \mid \mathsf{K} \mid \mathsf{C} \mid \mathsf{M} \mid (\mathsf{T},\mathsf{T}) \mid \{\mathsf{T}\}_{FuncName(\mathsf{T})},$$

where the '.' symbol is used to denote an empty term.

Having defined the terms exchanged by participants, we can proceed with the definition of a *node* and a *participant chain.* To capture the sending and receiving of terms, the definition of nodes uses *signed terms.* The occurrence of a term with a positive sign denotes transmission, while the occurrence of a term with a negative sign denotes reception.

**Definition 1.** *A* node *is any transmission or reception of a term denoted as* $\langle\sigma, t\rangle$*, with* $t \in \mathsf{T}$ *and* $\sigma$ *one of the symbols* $+, -$*. A node is written as* $-t$ *or* $+t$*. We use* $(\pm\mathsf{T})$ *to denote a set of nodes. Let* $n \in (\pm\mathsf{T})$*, then we define the function* $sign(n)$ *to map the sign and the function* $term(n)$ *to map the term corresponding to a given node.*

**Definition 2.** *A* participant chain *is a sequence of nodes. We use* $(\pm\mathsf{T})^*$ *to denote the set of finite sequences of nodes and* $\langle\pm t_1, \pm t_2, \ldots, \pm t_i\rangle$ *to denote an element of* $(\pm\mathsf{T})^*$*.*

In order to define a participant model we also need to define the preconditions that must be met such that a participant is able to execute a given protocol. In addition, we also need to define the effects resulting from a participant executing a protocol.

Preconditions and effects are defined using predicates applied on terms: *CON_TERM* : $\mathsf{T}$, denoting a generated term; *CON_PARTAUTH* : $\mathsf{T}$, denoting participant authentication; *CON_CONF* : $\mathsf{T}$, denoting the confidentiality of a given term); *CON_INTEG* : $\mathsf{T}$, denoting the integrity of a given term; *CON_NONREP* : $\mathsf{T}$, denoting the non-repudiation property for a given term; *CON_KEYEX* : $\mathsf{T}$, denoting a key exchange protocol.

The set of precondition-effect predicates is denoted by $\mathsf{PR\_CC}$ and the set of precondition-effect predicate subsets is denoted by $\mathsf{PR\_CC}^*$. The types attached to each protocol term are modeled using the following predicates: *TYPE_DN* : $\mathsf{T}$ to denote distinguished names, *TYPE_UD* : $\mathsf{T}$ to denote user-domain names, *TYPE_NT* : $\mathsf{T}$ to denote timestamps, *TYPE_NDH* : $\mathsf{T}$ to denote Diffie-Hellman random numbers, *TYPE_NA* : $\mathsf{T}$ to denote other random numbers, *TYPE_NDH* : $\mathsf{T} \times \mathsf{T} \times \mathsf{T} \times \mathsf{P} \times \mathsf{P}$ to denote Diffie-Hellman symmetric keys, *TYPE_KSYM* : $\mathsf{T} \times \mathsf{P} \times \mathsf{P}$ to denote symmetric keys, *TYPE_KPUB* : $\mathsf{T} \times \mathsf{P}$ to denote public keys, *TYPE_KPRV* : $\mathsf{T} \times \mathsf{P}$ to denote private keys, and *TYPE_CERT* : $\mathsf{T} \times \mathsf{P}$ do denote certificate terms.

The set of type predicates is denoted by $\mathsf{PR\_TYPE}$ and the set of type predicate subsets is denoted by $\mathsf{PR\_TYPE}^*$. Based on the defined sets and predicates we are now ready to define the participant and protocol models.

**Definition 3.** *A* participant model *is a tuple* $\langle prec, eff, type, gen, part, chain\rangle$, *where* $prec \in \mathsf{PR\_CC}^*$ *is a set of precondition predicates,* $eff \in \mathsf{PR\_CC}^*$ *is a set of effect predicates,* $type \in \mathsf{PR\_TYPE}$ *is a set of type predicates,* $gen \in \mathsf{T}^*$ *is a set of generated terms,* $part \in \mathsf{P}$ *is a participant name and* $chain \in (\pm\mathsf{T})^*$ *is a participant chain. We use the* $\mathsf{MPART}$ *symbol to denote the set of all participant models.*

**Definition 4.** *A* security protocol model *is a collection of participant models such that for each positive node* $n_1$ *there is exactly one negative node* $n_2$ *with* $term(n_1) = term(n_2)$. *We use the* $\mathsf{MPROT}$ *symbol to denote the set of all security protocol models.*

## 3 Semantic Security Protocol Model

In this section we propose a new semantic security protocol model (SSPM) based on which we construct security protocol specifications that can be automatically executed by protocol participants. The proposed model must maintain the security properties of the protocol and must provide sufficient information for participants to be able to execute the protocol.

Protocols are given using their SPM model described in the previous section. Based on this model we generate the corresponding SSPM that has two components: the sequential model (SEQM) and the ontology model (ONTM). The first component is implemented as a WSDL-S specification while the second component is implemented as an OWL specification. In the remaining of this section we provide a description of each component and we provide a set of rules to generate SSPM from a given SPM.

### 3.1 Sequential and Ontology Models

We use the symbol $\mathsf{URI}$ to denote the set of *Uniform Resource Identifiers*, $\mathsf{CONC}$ to denote the set of all concepts and $\mathsf{CONC}^*$ to denote the set of subsets with elements from $\mathsf{CONC}$.

**Definition 5.** *An* annotation *is a pair* $\langle uri, c\rangle$, *where* $uri \in \mathsf{URI}$ *and* $c \in \mathsf{CONC}$. *The set corresponding to a* SSPM *is denoted by* $\mathsf{ANNOT}$ *and the set of subsets with elements from* $\mathsf{ANNOT}$ *is denoted by* $\mathsf{ANNOT}^*$.

By consulting the WSDL-S specification we define a message as a pair consisting of the message direction and an annotation.

**Definition 6.** *A* message *is a pair* $\langle d, a\rangle$, *where* $d \in \{in, out\}$ *and* $a \in \mathsf{ANNOT}$. *We define* $\mathsf{MSG}$ *to denote a set of messages and* $\mathsf{MSG}^*$ *to denote the set of subsets with elements from* $\mathsf{MSG}$.

Next, we define the sequential model as a collection of preconditions, effects and messages, based on the previous definitions.

**Definition 7.** *A* sequential model *is a triplet* $\langle s\_prec, s\_eff, s\_msg \rangle$*, where* $s\_prec \in \mathsf{ANNOT}^*$ *is a set of preconditions,* $s\_eff \in \mathsf{ANNOT}^*$ *is a set of effects and* $s\_msg \in \mathsf{MSG}^*$ *is a set of messages.*

The ontology model follows the description of OWL.

**Definition 8.** *An* ontology model *is a triplet* $\langle conc, propr, inst \rangle$*, where* $conc \in \mathsf{CONC}$ *is a set of concepts,* $propr \in \mathsf{PROPR}$ *is a set of properties and* $inst \in \mathsf{INST}$ *is a set of instances. An element from propr is a pair* $\langle \alpha, \beta \rangle$*, where* $\alpha$ *is a unique id and* $\beta$ *is a syntactic construction denoting the property name.*

*Let* $pr_1 = \langle \alpha_1, \beta_1 \rangle$ *and* $pr_2 = \langle \alpha_2, \beta_2 \rangle$*. Then* $pr_1 = pr_2$ *iff* $\alpha_1 = \alpha_2$ *and* $\beta_1 = \beta_2$*. We define the function* $(\_)_{\mathrm{id}}$ *to map the* $\alpha$ *component and the function* $(\_)_{\mathrm{nm}}$ *to map the* $\beta$ *component of a given property.*

*We use* $\mathsf{PROPR}$ *to denote the set of all properties and* $\mathsf{INST}$ *to denote the set of all instances. We use* $\mathsf{PROPR}^*$ *to denote the set of all subsets with elements from* $\mathsf{PROPR}$ *and* $\mathsf{INST}^*$ *to denote the set of all subsets with elements from* $\mathsf{INST}$*.*

In order to handle the previously defined ontology model we define the function $(\_)_{\mathrm{d}} : \mathsf{PROPR} \rightarrow \mathsf{CONC}$ to map the domain concept of a given property, $(\_)_{\mathrm{c}} : \mathsf{PROPR} \rightarrow \mathsf{CONC}$ to map the category concept of a given property, $(\_, \_)_{\mathrm{ci}} : \mathsf{CONC} \times \mathsf{PROPR} \rightarrow \mathsf{INST}$ to map the instance corresponding to a domain concept and property, $(\_)^{\mathrm{s}}_{\mathrm{e}} : \mathsf{CONC} \rightarrow \mathsf{CONC}^*$ to map the set of concepts for which the given concept is parent, $(\_)_{\mathrm{p}} : \mathsf{CONC} \rightarrow \mathsf{PROPR}^*$ to map the set of properties for which the given concept is domain.

### 3.2 Generating the Semantic Security Protocol Model

In order to generate the SSPM for a given SPM, we start with a core ontology model (OM) (figure 1) that contains concepts found in classical security protocols. The core OM was constructed by consulting security protocols found in open libraries such as SPORE [15] or the library published by John Clark [5].

The core ontology is constructed from 7 sub-ontologies. The sub-ontologies that must be extended with new concepts for each SSPM are denoted in figure 1 by interrupted lines, while the permanent sub-ontologies are denoted by continuous lines. The new concepts are generated from SPM, however, information that is not available in the SPM must be provided by the user.

The *SecurityProperty* sub-ontology contains concepts such as *Authentication*, *Confidentiality* or *Session_key_exchange*. The *TermType* sub-ontology includes concepts related to term types used in security protocol messages such as *SymmetricKey*, *PublicKey* or *ParticipantName*. Concepts related to cryptographic specifications such as encryption algorithms or encryption modes are found in the sub-ontology *CryptoSpec*. In order to model modules needed to extract keys, names or certificates we use the *LoadingModule* sub-ontology. The *ParticipantRole* sub-ontology defines concepts modeling roles handled by protocol participants such as *Initiator*, *Respondent* and *Third_Party*.

The *Knowledge* sub-ontology contains 5 concepts: *PreviousTerm*, *AccessedModule*, *InitialTerm*, *GeneratedTerm* and *DiscoveredTerm*. Each concept defines a class of terms specific to security protocols: terms from previous executions, modules, initial terms, generated terms and discovered terms.

The last sub-ontology is *CommunicationTerm*, which defines two concepts: *SentTerm* and *ReceivedTerm*. This sub-ontology is extended for each SEM-S with concepts that are sent or received. For each concept, functional properties are defined denoting the operations performed on the terms corresponding to concepts. The concepts used to extend the core ontology are specific to each

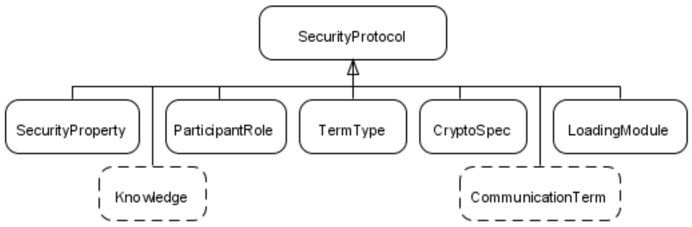

Fig. 1: Core ontology of SSPM

protocol, however, the defined properties are applied on all constructions. From these properties we mention: *hasKey*, *isStored*, *isVerified*.

In order to generate the SSPM from a given SPM we define a set of rules and generating algorithms. The developed rules use the $\_ \leftarrow_{\mathrm{r}} \_$ operator to denote set reunion and the $\_ \leftarrow_{\mathrm{a}} \_$ operator to denote a value transfer.

The first two rules generate the predicate concepts corresponding to preconditions *prec* from a SPM, where the function $gc : \mathsf{T} \rightarrow \mathsf{CONC}$ is used to generate the concept corresponding to a given term and the function $gcc : \mathsf{PR\_CC} \rightarrow \mathsf{CONC}$ is used to generate the concept corresponding to a given precondition predicate:

$$\frac{pr \in prec \;\; pr = CON\_TERM(t)}{c \leftarrow_{\mathrm{a}} gc(t) \;\; s\_prec \leftarrow_{\mathrm{r}} \{\langle uri, c\rangle\} \;\; (InitialTerm)^{\mathrm{s}}_{\mathrm{e}} \leftarrow_{\mathrm{r}} \{c\}}\text{pr\_term},$$

$$\frac{pr \in prec \;\; pr \neq CON\_TERM(t)}{s\_prec \leftarrow_{\mathrm{r}} \{\langle uri, gcc(pr)\rangle, \langle uri, gc(t)\rangle\}}\text{pr\_propr}.$$

The rules generating the effects have a similar structure because of the *eff* set. For each positive or negative node there is a corresponding concept in the *SentTerm* and *ReceivedTerm* sub-ontologies, generated by the following rules:

$$\frac{n \in chain \;\; sign(n) = +}{c \leftarrow_{\mathrm{a}} gtx(term(n)) \;\; s\_msg \leftarrow_{\mathrm{r}} \{\langle out, c\rangle\} \;\; (SentTerm)^{\mathrm{s}}_{\mathrm{e}} \leftarrow_{\mathrm{r}} \{c\}}\text{msg\_tx},$$

$$\frac{n \in chain \;\; sign(n) = -}{c \leftarrow_{\mathrm{a}} grx(term(n)) \;\; s\_msg \leftarrow_{\mathrm{r}} \{\langle in, c\rangle\} \;\; (ReceivedTerm)^{\mathrm{s}}_{\mathrm{e}} \leftarrow_{\mathrm{r}} \{c\}}\text{msg\_rx}.$$

The concatenated terms corresponding to each transmitted or received term are modeled using similar rules. For each sent term the SSPM must provide the construction operations and for each received term the SSPM must provide processing operations. Sub-concepts of *SentTerm* are connected to sub-concepts of *Knowledge* through the *isExtracted* property, generated according to the following rule, where we used the function $\mathsf{PR\_CC}^* \rightarrow \mathsf{ID}$ to generate a new property id:

$$\frac{c \in (\mathit{SentTerm})^{\mathrm{s}}_{\mathrm{e}}}{p \leftarrow_{\mathrm{a}} \langle \mathit{gid}(\mathit{propr}), \mathit{isExtracted}\rangle \;\; (c)_{\mathrm{p}} \leftarrow_{\mathrm{r}} \{p\} \;\; (p)_{\mathrm{c}} \in (\mathit{Knowledge})^{\mathrm{s}}_{\mathrm{e}}}\text{con\_extr.}$$

Processing of received terms is done according to the type of the given term and to the knowledge available to the user. The modeled operations introduce constraints on the type and location of knowledge through the following rules, where we used the $E\_SYM : \mathsf{CONC}$ predicate to denote symmetric encryption:

$$\frac{c \in (\mathit{ReceivedTerm})^{\mathrm{s}}_{\mathrm{e}} \;\; p \in (c)_{\mathrm{p}} \;\; (p)_{\mathrm{nm}} = \mathit{isDecrypted}}{c' \leftarrow_{\mathrm{a}} (p)_{\mathrm{c}} \;\; E\_SYM(c') \vee E\_SYM(c') \;\; (c') \in (\mathit{DiscoveredTerm})^{\mathrm{s}}_{\mathrm{e}}}\text{con\_decr,}$$

$$\frac{c \in (\mathit{ReceivedTerm})^{\mathrm{s}}_{\mathrm{e}} \;\; p \in (c)_{\mathrm{p}} \;\; (p)_{\mathrm{nm}} = \mathit{isStored}}{c' \leftarrow_{\mathrm{a}} (p)_{\mathrm{c}} \;\; (c') \in (\mathit{DiscoveredTerm})^{\mathrm{s}}_{\mathrm{e}}}\text{con\_stored,}$$

$$\frac{c \in (\mathit{ReceivedTerm})^{\mathrm{s}}_{\mathrm{e}} \;\; p \in (c)_{\mathrm{p}} \;\; (p)_{\mathrm{nm}} = \mathit{isVerified}}{c' \leftarrow_{\mathrm{a}} (p)_{\mathrm{c}} \;\; (c') \in \{(\mathit{DiscoveredTerm})^{\mathrm{s}}_{\mathrm{e}} \setminus (\mathit{AccessedModule})^{\mathrm{s}}_{\mathrm{e}}\}}\text{con\_verif.}$$

In the *Knowledge* sub-ontology, each concept has an *isOfType* property attached based on which participants can decide on the operations to execute. For each type, additional properties are defined such as the *hasSymmAlg* or *hasKey* properties for symmetric encrypted terms. The rules based on which these properties are generated are specific to each type. For example, the following rules define the algorithm type and key for an encrypted term that must be processed or constructed:

$$\frac{c \in (\mathit{Knowledge})^{\mathrm{s}}_{\mathrm{e}} \;\; E\_SYM(c)}{p \leftarrow_{\mathrm{a}} \langle \mathit{gid}(\mathit{propr}), \mathit{hasSymmAlg}\rangle \;\; (c)_{\mathrm{p}} \leftarrow_{\mathrm{r}} \{p\} \;\; (c,p)_{\mathrm{ci}} \in (\mathit{Symmetric})^{\mathrm{s}}_{\mathrm{e}}}\text{sim\_alg,}$$

$$\frac{c \in (\mathit{Knowledge})^{\mathrm{s}}_{\mathrm{e}} \;\; E\_SYM(c)}{p \leftarrow_{\mathrm{a}} \langle \mathit{gid}(\mathit{propr}), \mathit{hasKey}\rangle \;\; (c)_{\mathrm{p}} \leftarrow_{\mathrm{r}} \{p\} \;\; (p)_{\mathrm{c}} \in (\mathit{Knowledge})^{\mathrm{s}}_{\mathrm{e}}}\text{sim\_key.}$$

The rules presented above are executed by algorithms. For example, modeling positive nodes in SSPM is done through the use of algorithm 1. Here, the set of knowledge *KNOW* corresponding to each executing participant grows with the construction and reception of each new term. We used the function $\mathit{mpart} : \mathsf{T} \rightarrow \mathsf{T}^*$ to map the set of concatenated terms and the keyword "Exec" to denote the execution of sub-algorithms.

### 3.3 Correctness of SSPM

In the generation process of SSPM from a given SPM, we consider a correct SPM constructed by the user. With the large number of attacks reported in the

**Algorithm 1** Model positive and negative nodes

```
Require: $n \in (\pm\mathsf{T}), sign(n) = +$
  for all $t \in mpart(term(n))$ do
    Let $c = gc(t)$
    Let $p \Leftarrow$ @con_extr(c)
    if $t \in KNOW$ then
      $(p)_{\mathrm{c}} \leftarrow_{\mathrm{a}} c$
    else if $t = \{t'\}_{f(k)}$ then
      $(GeneratedTerm)^{\mathrm{s}}_{\mathrm{e}} \leftarrow_{\mathrm{r}} \{c\}$
      Exec $ModelEncryptedGenerated(t)$
    else if $t \in gen$ then
      $(GeneratedTerm)^{\mathrm{s}}_{\mathrm{e}} \leftarrow_{\mathrm{r}} \{c\}$
      Exec $ModelPlainGenerated(t)$
    else
      $(DiscoveredTerm)^{\mathrm{s}}_{\mathrm{e}} \leftarrow_{\mathrm{r}} \{c\}$
      Exec $ModelDiscoveredLoaded(t)$
    end if
    $KNOW \leftarrow_{\mathrm{r}} t$
  end for
```

literature [6], [7], it is vital for new protocol models to maintain the security properties of protocols for which security properties have been proved to hold.

In order to prove the correctness of the generated SSPM, we consider $\Gamma$ representing the set of all information included in an SSPM. The information generated by the proposed rules can be divided into three components: mapped information, user-provided information and participant knowledge-based information.

The set of mapped information is denoted by $\gamma_{map}$ and represents information originating directly from SPM. The set of user-provided information is denoted by $\gamma_{up}$ and represents information originating from the user (e.g. cryptographic algorithms). The set of knowledge-based information originates from the knowledge available when running the protocol and is denoted by $\gamma_{know}$.

By using the above sets $\Gamma = \gamma_{map} \cup \gamma_{up} \cup \gamma_{know}$. The correctness of the information contained in $\gamma_{map}$ results from the original protocol model, while the correctness of the information contained in $\gamma_{up}$ results from the assumption that the user provides correct information.

The information contained in $\gamma_{know}$ is generated based on the design principles of *fail-stop* [11] protocols. These principles state that the correctness of each received term must be verified and the protocol execution must be stopped immediately in case of invalid terms. By using these principles, the rules we proposed generate verification properties for each received term found in the participant's knowledge set. Protocols that do not follow these rules can not be modeled with our method.

The the correctness of the generated SSPM follows from the correctness of the information generated in the $\Gamma$ set, constructed from the three sets $\gamma_{map}, \gamma_{up}, \gamma_{know}$ for which the correctness has been discussed above.

## 4 Experimental Results

In this section we exemplify the construction of a SSPM from a given SPM and provide a few experimental result from implementing several generated SSPM.

### 4.1 Constructing the SSPM for the "BAN" protocol

In order to provide an example for constructing an SSPM for a given SPM, we use the well-known "BAN Concrete Secure Andrew RPC" protocol [15]. This is a two-party protocol providing a session key exchange using symmetric cryptography. The protocol assumes that participants are already in the possession of a long-term key $K_{ab}$.

Because of space considerations, we only provide the construction of the SSPM for the $A$ participant. Based on this, the construction of the SSPM for the second participant is straight-forward.

The precondition set $prec_A$ for participant $A$ is $prec_A = \{CON\_TERM(A), CON\_TERM(B), CON\_TERM(K_{ab})\}$ and the effect set $eff_A$ for the same participant is $eff_A = \{CON\_KEYEX(K_{ab})\}$. The set $type_A = \{TYPE\_UD\ (A), TYPE\_UD(B), TYPE\_KSYM(A, B, K_{ab}), TYPE\_KSYM(A, B, K), TYPE\_NA\ (N_a), TYPE\_NA(N_b)\}$ defines the type corresponding to each term and the set $gen_A = \{N_a\}$ defines the terms generated by participant $A$. The participant name is $part_A = A$ and the participant chain is $chain_A = \langle +(A, N_a), -\{N_a, K, B\}_{sk(K_{ab})}, +\{N_a\}_{sk(K)}, -N_b\rangle$.

By applying the rules and algorithms described in the previous sections we generate the SSPM model. Due to space considerations, instead of describing the actual SSPM we describe the implementation of the model. The sequential model is implemented as a WSDL-S specification, while the ontology model is implemented as a OWL specification.

Part of the resulted WSDL-S specification is given in figure 2 and part of the graphical representation of the OWL specification is given in figure 3.

```
...
 <xsd:element name="Msg1Request">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="Term1" type="xsd:base64Binary"
                    wssem:modelReference=".../SecProt.owl#SentTerm1">
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
 ...
 <wsdl:operation name="Msg1">
   <wsdl:output message="tns:Msg1Request"/>
 </wsdl:operation>
 <wssem:effect name="SessionKeyExchange"
               wssem:modelReference=".../SecProt.owl#SessionKey"/>
 ...
```

Fig. 2: Part of the sequential model's implementation

### 4.2 Executing the Generated Specifications

In order to prove that the SSPM model contains sufficient information for participants to execute the generated specifications, we generated several WSDL-S and OWL specifications corresponding to initiator and respondent protocol roles.

In order to execute the specifications, messages were encoded and transmitted according to the constructions provided by the WS-Security standard [17]. In the experiments we conducted, participants downloaded the specification files from a public server and they were able to execute the protocols based only on the received descriptions. The participants hardware and software configurations: Intel Dual Core CPU at 1.8GHz, 1GByte of RAM, MS Windows XP.

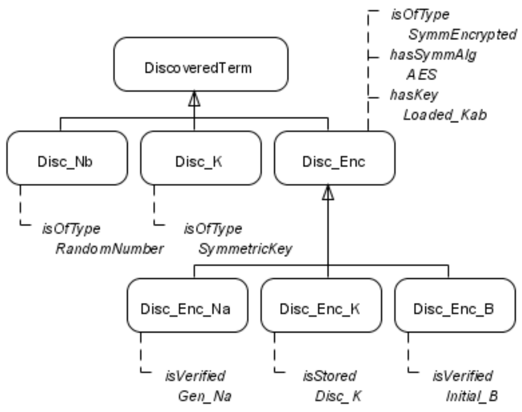

Fig. 3: Part of the ontology model's implementation

Part of the experimental results are given in table 1, where the values correspond to milliseconds. The "Spec. proc" column denotes the specification processing time, the "Msg. constr." column denotes the message construction time (for output messages) and the "Msg. proc." column denotes the message processing time (for input messages). The table contains two two-party protocols ("BAN Concrete Andrew Secure RPC", or more simply BAN, and ISO9798) and one three-party protocol (Kerberos). The performance differences between the BAN and ISO9798 protocols are due to the fact that ISO9798 makes use of public key cryptography, while BAN uses only symmetric cryptography.

## 5 Related Work

In this section we describe approaches we found in the literature that mostly relate to our proposal.

An approach that aims at the automatic implementation of security protocols is given in [2]. This approach uses a formal description as a specification which

Table 1: Protocol execution timings

| **Protocol participant** | **Spec. proc. (ms)** | **Msg. constr. (ms)** | **Msg. proc. (ms)** | **Total (ms)** |
|---|---|---|---|---|
| BAN Init. | 14.58 | 11.81 | 3.68 | 30.08 |
| BAN Resp. | 14.03 | 2.86 | 1.62 | 18.52 |
| ISO9798 Init. | 13.07 | 35.784 | 23.30 | 72.16 |
| ISO9798 Resp. | 13.51 | 6.876 | 12.24 | 32.63 |
| Kerb. Init. 1 | 22.63 | 0.83 | 0 | 23.47 |
| Kerb. Init. 2 | 12.61 | 0.55 | 1.58 | 14.76 |
| Kerb. Init. 3 | 2.23 | 3.34 | 0.94 | 6.52 |
| Kerb. Resp. 1 | 19.28 | 0 | 0.41 | 19.69 |
| Kerb. Resp. 2 | 10.81 | 3.379 | 1.67 | 15.87 |
| Kerb. Resp. 3 | 5.25 | 11.41 | 3.59 | 20.26 |

is executed by participants. The proposed specification does not make use of Web service technologies, because of which inter-operability and extendability of systems executing the given specifications becomes a real issue.

Abdullah and Menasc propose in [3] a specification that is constructed as an XML document from which code is generated. The resulted code is then compiled and executed by participants. Because of this aspect, our proposal is more dynamic in the sense that applications can download and execute new protocols based on the developed specifications automatically, without having to stop program execution.

The authors from [8] propose a security ontology for resource annotation. The proposed ontology defines concepts for security and authorization, for cryptographic algorithms and for credentials. This proposal was designed to be used in the process of security protocol description and selection based on several criteria. In contrast, our ontologies, have a more detailed construction. For example, the ontology from [8] defines a collection of cryptographic algorithms, however, it does not define the algorithm mode, which is an implementation-specific information.

There have been several other security ontologies proposed [9], [10] which can be used to complete our core ontology with additional concepts and properties, for generating more complex protocol models.

## 6 Conclusion and Future Work

We developed a novel method for the automated execution of security protocols. Our approach is based on a semantic security protocol model from which security protocol specifications are generated. The sequential component of the proposed model is implemented as a WSDL-S specification while the ontology component is implemented as an OWL specification.

Constructing the SSPM model is not a trivial task and can induce new flaws in correct protocols that can lead to attacks. In order to ensure a correct construction process, we developed several generating rules and algorithms that map each component from the input protocol model to a component in the SSPM model.

As future work we intend to develop a service-based middleware to support secure distribution of these specifications. The middleware will also be able to create new protocols based on already existing protocols and distribute the new specifications to Web services.

## References

1. Cremers, C.J.F., Mauw. S.: Checking secrecy by means of partial order reduction. In Leue S., Systa, T. (eds.) (2003), revised selected papers LNCS, Vol. 3466, Springer (2005).
2. Mengual, L., Barcia, N., Jimenez, E., Menasalvas, E., Setien, J., Yaguez, J: Automatic implementation system of security protocols based on formal description techniques. Proceedings of the Seventh International Symposium on Computers and Communications, pp. 355–401 (2002).
3. Abdullah, I., Menasc, D: Protocol specification and automatic implementation using XML and CBSE. IASTED conference on Communications, Internet and Information Technology (2003).
4. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Verma, K.: Web Service Semantics - WSDL-S. A joint UGA-IBM Technical Note (2005).
5. Clark, J., Jacob, J.: A Survey of Authentication Protocol Literature: Version 1.0. York University (1997).
6. Gavin, L.: Some new attacks upon security protocols. In Proceedings of the 9th CSFW, IEEE Computer Society Press, pp. 162–169, (1996).
7. Cremers, C.J.F.: Compositionality of Security Protocols: A Research Agenda. Electr. Notes Theor. Comput. Sci., Vol. 142, pp. 99–110 (2006).
8. Kim, A., Luo, J., Kang, M.: Security ontology for annotating resources. Lecture Notes In Computer Science, Vol. 3761, pp. 1483–1499 (2005).
9. Blanco, C., Lasheras, J., Valencia-Garcia, R., Fernandez-Medina, E., Toval, A., Piattini, M.: A systematic review and comparison of security ontologies. Proc. of the Third International Conference on Availability, Reliability and Security, pp. 813–820 (2008).
10. Denkera, G., Kagal, L., Finin, T.: Security in the semantic web using owl. Information Security Technical Report, Vol. 1(10), pp. 51–58 (2005).
11. Gong, L.: Fail-Stop Protocols: An Approach to Designing Secure Protocols. In Proceedings of the 5th IFIP Conference on Dependable Computing and Fault-Tolerant Systems, pp. 44–55 (1995).
12. World Wide Web Consortium, OWL Web Ontology Language Reference, W3C Recommendation (2004).
13. Gutmann, P.: Cryptlib Encryption Toolkit. http://www.cs.auckland.ac.nz/-pgut001/cryptlib/index.html.
14. OpenSSL Project, version 0.9.8h, http://www.openssl.org/.
15. Laboratoire Specification et Verification, Security Protocol Open Repository, http://www.lsv.ens-cachan.fr/spore.
16. Organization for the Advancement of Structured Information Standards, SAML V2.0 OASIS Standard Specification, http://saml.xml.org/ (2007).
17. Organization for the Advancement of Structured Information Standards, OASIS Web Services Security (WSS), http://saml.xml.org/ (2006).