Automated Composition of Security Protocols

Genge Béla
"Petru Maior" University of Târgu Mureş
Electrical Engineering Department
N. Iorga St., No. 1, Tg. Mureş, Romania
bgenge@engineering.upm.ro

Iosif Ignat
Technical University of Cluj-Napoca
Computer Science Department
Gh. Baritiu St., No. 28, Cluj-Napoca, Romania
Iosif.Ignat@cs.utcluj.ro

Haller Piroska
"Petru Maior" University of Târgu Mureş
Electrical Engineering Department
N. Iorga St., No. 1, Tg. Mureş, Romania
phaller@engineering.upm.ro

Abstract

Determining if two protocols can be securely composed requires analyzing not only their additive properties but also their destructive properties. In this paper we propose a new composition method for constructing protocols based on existing ones found in the literature that can be fully automatized. The additive properties of the composed protocols are ensured by the composition of protocol preconditions and effects, denoting, respectively, the conditions that must hold for protocols to be executed and the conditions that hold after executing the protocols. The non-destructive property of the final composed protocol is verified by analyzing the independence of the involved protocols, a method proposed by the authors in their previous work. The fully automatized property is ensured by constructing a rich protocol model that contains explicit description of protocol preconditions, effects, generated terms and exchanged messages. The proposed method is validated by composing 17 protocol pairs and by verifying the correctness of the composed protocols with an existing tool.

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. Achieving such security goals is made through the use of cryptography. The explosive development of today's Internet and the technological advances made it possible to implement and use security protocols in a wide range of applications such as sensor networks, electronic commerce or routing environments.

Security protocols have been intensively analyzed throughout the last few decades, resulting in a variety of dedicated formal methods and tools [2, 3, 4]. The majority of these methods consider a Dolev-Yao-like intruder model [5, 6] to capture the actions available to the intruder that has complete control over the network. By analyzing each individual protocol in the presence of this intruder, the literature has reported numerous types of attacks [3, 7]. However, in practice, there can be multiple protocols running over the same network, thus the intruder is given new opportunities to construct attacks by combining messages from several protocols, also known as multi-protocol attacks [8].

Designing new protocols, thus, becomes a challenging task if we look at the number of attacks that have been discovered over the years [7] after the protocols have been published. In the last few years the use of protocol composition [8, 9, 10] has been successfully applied to create new protocols based on existing [11, 12, 13] or predefined protocols [9].

In this paper we propose a new composition method that, as opposed to existing approaches [9, 11, 12, 13, 14] can be fully automatized by eliminating the human factor. In order to create an automated composition method, we need an enriched protocol model that contains enough information to compose the protocol preconditions and effects and an approach for the verification of the correctness of the final, composed protocol.

Preconditions denote the set of properties that must be satisfied for the protocol to be executed, while the effects denote the set of properties resulting from the protocol execution. By composing preconditions and effects (i.e. PE composition), we generate a new protocol sequence that ensures the satisfaction of the protocol preconditions and the propagation of generated information through effects.

The protocol sequence generated by the PE composition must be correct, in the sense that it must maintain the security properties of the original protocols. In order to verify this, we use an approach developed in our previous work [15] that verifies the independence of the involved protocols. Protocol independence, called participant chain composition (i.e. PC composition) ensures that the intruder can not replay messages from one protocol to another to construct new attacks while running the protocols in the same environment. This property also ensures the correctness of the composed protocol.

The paper is structured as follows. In section 2 we define an enriched protocol model that includes explicit description of protocol preconditions, effects, generated terms and exchanged messages. In section 3 we provide a description of the proposed composition method and a brief presentation of the independence verification method proposed in our previous work [15]. The proposed composition method has been applied in the composition process of several protocols, part of these experimental results are given in section 4. We relate our work to others found in the literature in section 5 and 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: P, denoting the set of participant names; N, denoting the set of random numbers or *nonces* (i.e. "number once used"); K, denoting the set of cryptographic keys; C, denoting the set of certificates and 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 [17, 18] 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;
- N_T, denoting the set of timestamps; 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 σ 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* : T, denoting a term that must be previously generated (preconditions) or it is generated (effects); *CON_PARTAUTH* : T, denoting a participant that must be previously authenticated (preconditions) or a participant that is authenticated (effects); *CON_CONF* : T, denoting that a given term must be confidential (preconditions) or it is kept confidential (effects); *CON_INTEG* : T, denoting that for a given term the integrity property must be provided (preconditions) or that the protocol ensures the integrity property for the given term (effects); *CON_NONREP* : T, denoting that for a given term the non-repudiation property must be provided (preconditions) or that the protocol ensures the non-repudiation property for the given term (effects); *CON_KEYEX* : T, denoting that a key exchange protocol must be executed before (preconditions) or that this protocol provides a key exchange resulting the given term (effects).

The set of precondition-effect predicates is denoted by $\mathsf{PR\_CC}$ and the set of precondition-effect predicate subsets is denoted by $\mathsf{PR\_CC}^*$. Next, we define predicates for each type of term exchanged by protocol participants. These predicates are based on the basic and specialized sets provided at the beginning of this section. We use the *TYPE_DN* : T predicate to denote distinguished name terms, *TYPE_UD* : T to denote user-domain name terms, $TYPE\_IP$: T to denote user-ip name terms, *TYPE_U* : T user name terms, *TYPE_NT* : T to denote timestamp terms, *TYPE_NDH* : T to denote Diffie-Hellman random number terms, *TYPE_NA* : T to denote other random number terms, *TYPE_NDH* : $\mathsf{T} \times \mathsf{T} \times \mathsf{T} \times \mathsf{P} \times \mathsf{P}$ to denote Diffie-Hellman symmetric key terms $(term, number_1, number_2, participant_1, participant_2)$, *TYPE_KSYM* : $\mathsf{T} \times \mathsf{P} \times \mathsf{P}$ to denote symmetric key terms $(term, participant_1, participant_2)$, *TYPE_KPUB* : $\mathsf{T} \times \mathsf{P}$ to denote public key terms $(term, participant)$, *TYPE_KPRV* : $\mathsf{T} \times \mathsf{P}$ to denote private key terms $(term, participant)$, *TYPE_CERT* : $\mathsf{T} \times \mathsf{P}$ do denote certificate terms $(term, participant)$ and *TYPE_MSG* : T to denote user-defined 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* $\langle prec, \mathit{eff}, type, gen, part, chain\rangle$*, where* $prec \in \mathsf{PR\_CC}^*$ *is a set of precondition predicates,* $\mathit{eff} \in \mathsf{PR\_CC}^*$ *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* MPART *symbol to denote the set of all participant models.*

Definition 4. *A* 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* MPROT *symbol to denote the set of all protocol models.*

3 Composition of protocol models

The composition process involves composing in a first stage the protocol preconditions and effects followed by the composition of participant chains. In this section we first formulate the conditions needed for the precondition-effect (PE) composition which involves establishing the satisfaction of protocol preconditions and the verification of the non-destructive properties of protocol effects. This is followed by the protocol-chain (PC) composition for which we construct a canonical model and verify the independence of the involved participant chains.

3.1 Composition of preconditions and effects

In the composition process of two security protocols we first need to compose the preconditions and effects. In other words, we need to establish if the knowledge needed by protocol participants to run a given protocol, expressed through the form of precondition predicates, is available and if the set of precondition and effect predicates is non-destructive.

In order to establish if the set of preconditions corresponding to a protocol can be satisfied based on a given context and the effects corresponding to another protocol we use the predicate *PART_PREC* : $\mathsf{T}^* \times \mathsf{PR\_CC}^* \times \mathsf{PR\_CC}^*$. The context denotes the initial knowledge available to participants when running the protocol. For two participant models, $\varsigma_1 = \langle prec_1, \mathit{eff}_1, type_1, gen_1, part_1, chain_1\rangle$ and $\varsigma_2 = \langle prec_2, \mathit{eff}_2, type_2, gen_2, part_2, chain_2\rangle$, the *PART_PREC* predicate is defined as

$$\mathit{PART\_PREC}(ctx, \mathit{eff}_1, prec_2) = \begin{cases} True, & \text{if } \mathit{eff}_1 \subseteq prec_2 \cup \text{ ,} \\ & \quad \{\cup\{CON\_TERM(t) | t \in ctx\}\} \text{ ,} \\ False, & \text{otherwise .} \end{cases}$$

The non-destructive property applies only for the *CON_CONF* because the absence of another property, such as integrity or non-repudiation, does not affect the previous properties. In order to establish if the preconditions and effects of two participant models are destructive we use the predicate *PART_NONDESTR* : $\mathsf{PR\_CC}^* \times \mathsf{PR\_CC}^* \times \mathsf{PR\_CC}^*$ which holds only if all confidential terms from one participant model maintain their confidentiality property in the second participant model also. Thus, the predicate is defined as

$$\mathit{PART\_NONDESTR}(\mathit{eff}_1, prec_2, \mathit{eff}_2) = \begin{cases} True, & \text{if } EF_1 \neq \mathit{CON\_CONF} \vee \\ & \quad \text{if } EF_1 = \mathit{CON\_CONF} \wedge t_1 = t_2 \text{ then} \\ & \quad\quad \exists EF_2(t_2) : EF_2 = \mathit{CON\_CONF}, \\ & \forall EF_1(t_1) \in \mathit{eff}_1 \wedge \forall PR_2(t_2) \in prec_2, \\ False, & \text{otherwise.} \end{cases}$$

Based on the above given predicates we can state that in order to compose the preconditions and effects corresponding to two participant models we need to establish if the predicates *PART_PREC* and *PART_NONDESTR* hold. The precondition-effect (PE) composition is expressed through the use of the operator $\_ \prec_{\varsigma}^{PE} \_ : \mathsf{MPART} \times \mathsf{MPART} \rightarrow \mathsf{MPART}$, which generates a new participant model based on two given participant models. By using this operator, we not only express the PE composition of participant models but also the order in which the given participant models appear in the final, composed participant model. Thus, we can state that given two participant models, ς_1 and ς_2, for which the PE composition requirements are satisfied, we have that $\varsigma_1 \prec_{\varsigma}^{PE} \varsigma_2 \neq \varsigma_2 \prec_{\varsigma}^{PE} \varsigma_1$. If the operator is applied on two participant models that can not be composed (i.e. one of the two predicates does not hold), the result is the empty participant model, denoted by $\phi_{\varsigma} = \langle \phi, phi, \phi, \phi, ., \langle\rangle \rangle$, where ϕ denotes an empty set.

The PE composition requirements of two participant models can be easily extended to form the requirements for the PE composition of two protocol models. These requirements include applying the $\_ \prec_{\varsigma}^{PE} \_$ operator on pairs of participant models for which the names are equal. We express the PE composition of two protocol models through the use of the $\_ \prec_{\xi}^{PE} \_ : \mathsf{MPROT} \times \mathsf{MPROT} \rightarrow \mathsf{MPROT}$ operator. For this operator also, we can state that given two protocol models, ξ_1 and ξ_2, for which the PE composition requirements are satisfied, we have that $\xi_1 \prec_{\xi}^{PE} \xi_2 \neq \xi_2 \prec_{\xi}^{PE} \xi_1$. In case of protocol models that can not be composed, the result is denoted by the empty protocol model $\phi_{\xi} = \phi$.

3.2 Composition of participant chains

The PC composition makes use of a canonical model that focuses on terms that can be verified by protocol participants. For each term the canonical model provides a corresponding syntactical representation through the use of *basic types*. These denote the terms that can be verified by protocol participants also including a representation for terms that can not be verified because of limited participant knowledge. The verification process makes use of these types to decide if attacks can be constructed on each protocol model by using terms extracted from the other considered protocol models.

In order to compose two participant chains these must be *instance independent* and *canonical independent*. The first condition refers to the non-destructive properties of preconditions and effects while the second condition refers to verifying the independence of the involved participant chains based on the canonical model. The verification of the independence property of protocol models has been covered by the authors in their previous work [15]. If protocols are independent, then they maintain their security properties when they are run in the same context. By using this property in the composition process, protocols maintain their security properties, resulting new protocols with accumulated properties.

In the remaining of this section we briefly present the canonical model and the protocol independence property proposed in our previous work.

The *basic types* we consider are based on the specialized basic sets introduced in the protocol model:

$$\begin{aligned} BasicType ::= \ & \mathsf{p}_{DN} \mid \mathsf{p}_{UD} \mid \mathsf{p}_{IP} \mid \mathsf{p}_{U} \mid \mathsf{n}_{T} \mid \mathsf{n}_{DH} \\ & \mid \mathsf{n}_{A} \mid \mathcal{K} \mid \mathsf{m} \mid \mathsf{c} \mid \mathsf{u}, \end{aligned}$$

where the given symbols correspond to participant distinguished names, user-domain names, user-ip names, other user names, timestamps, Diffie-Hellman random numbers, other random numbers, keys, user defined terms, certificates and unknown terms, respectively.

The *unknown* type u corresponds to terms that can not be validated because of limited participant knowledge. By including this information in the specification we are able to detect subtle type-flaw attacks using a syntactical comparison of typed terms, that otherwise would require the construction of a state-space that can become rather large if we consider the existence of multiple protocols in the same system [16].

Based on the defined basic terms we can now proceed with the definition of *canonical terms*:

$$\mathcal{T} ::= . \mid BasicType \mid (\mathcal{T}, \mathcal{T}) \mid \{\mathcal{T}\}_{FuncName(\mathcal{T})}.$$

A canonical node is defined as a signed canonical term using the following definition.

Definition 5. *A* canonical node *is any transmission or reception of a canonical term denoted by $\langle \sigma, t \rangle$, with $t \in \mathcal{T}$ and σ one of the symbols $+, -$. We use $(\pm\mathcal{T})$ to denote a set of canonical nodes. Let $n \in (\pm\mathcal{T})$, then we define the function $csign(n)$ to map the sign and the function $cterm(n)$ to map the canonical term corresponding to a given canonical node.*

Before we proceed with the definition of canonical chains and canonical participant models we need to define *classifiers*. These are attached to participant chains and are used to transform canonical terms received from other participants based on local participant knowledge. We define two such classifiers:

$$Classifier ::= CL_P \mid CL_V.$$

The first classifier CL_P denotes the processing chain corresponding to a participant. This chain contains canonical terms that correspond to participant knowledge. The second classifier CL_V denotes the virtual chain used to transform received terms from the transmitted form to the received form based on the knowledge of the receiving participant.

Definition 6. *A* canonical participant chain *is a sequence of canonical nodes. A* classified canonical participant chain *is a pair $\langle CL, l_{cc}\rangle$, where $CL \in Classifier$ and $l_{cc} \in (\pm\mathcal{T})^*$. We use $(\pm\mathcal{T})^*$ to denote a set of canonical participant chains.*

Definition 7. *A* canonical participant model *is a pair $\langle part, sl_{cc}\rangle$, where $part \in$ P is a participant name and $sl_{cc} \in (Classifier \times (\pm\mathcal{T})^*)^*$ is a set of classified canonical participant chains. We use* MPART-C *to denote the set of all canonical participant models.*

Next, we define a canonical protocol model as a set of canonical participant models.

Definition 8. *A* canonical protocol model *is a collection of canonical participant models such that for each positive canonical node n_1 there is exactly one negative canonical node n_2 with $cterm(n_1) = cterm(n_2)$. We use the* MPROT-C *symbol to denote the set of all canonical protocol models.*

Based on the described protocol and canonical models, we proved, through the form of a proposition, that if two protocol models are instance independent and their corresponding canonical models are canonical independent, then the intruder can not construct attacks using terms extracted from other protocols. In order to verify this we used an intruder model based on the Dolev-Yao [5, 6] model to capture the powers that can be used by an intruder.

If two protocol models are independent, then their participant chains can be composed. We use the $\_ \prec^{PC}_{\varsigma} \_ :$ MPART × MPART → MPART operator to denote the PC composition of protocol chains and the $\_ \prec^{PC}_{\xi} \_ :$ MPROT × MPROT → MPROT operator to denote the PC composition of protocol models. For the first operator we use ϕ_ς to denote the empty participant model, while for the second operator we use ϕ_ξ to denote the empty protocol model.

If two protocol models can be composed PE and PC, then they can be composed. The composition operator we use to denote the composition of protocol models is $\_ \prec^{C} \_ :$ MPROT × MPROT → MPROT, for which the generated empty protocol model is denoted by ϕ_ξ.

By sequentially composing several protocol models the resulting protocol model provides a unified set of preconditions and effects and a unified set of participant chains. By composing *i* protocols, the resulting sequence is written as $\xi_1 \prec^C \xi_2 \prec^C \ldots \prec^C \xi_i$.

3.3 Composition algorithm

The proposed composition method can be applied on protocol pairs or entire protocol sequences. Let SEQ_1 and SEQ_2 be two protocol sequences, where each sequence is constructed by subsequently applying the $\_ \prec^C \_$ operator on protocol pairs, and n, m, two symbols denoting the number of protocols in the first and in the second sequence, respectively. Then, the composition algorithm must ensure that the new composed sequence maintains the security properties of the original protocols and that the knowledge available to protocol participants allows the execution of the new sequence. Verifying if protocols from the two

Algorithm 1 Composition steps

```
{Verification of non-destructive properties}
for all $\xi_1 \in SEQ_1$ and $\xi_2 \in SEQ_2$ do
  for all $\varsigma_1 \in \xi_1$ and $\varsigma_2 \in \xi_2$ do
    Let $\varsigma_1 = \langle prec_1, \mathit{eff}_1, type_1, gen_1, part_1, chain_1\rangle$,
        $\varsigma_2 = \langle prec_2, \mathit{eff}_2, type_2, gen_2, part_2, chain_2\rangle$,
        $c_1 = \mathit{PART\_NONDESTR}(\mathit{eff}_1, prec_2, \mathit{eff}_2)$,
        $c_2 = \mathit{PART\_NONDESTR}(\mathit{eff}_2, prec_1, \mathit{eff}_1)$
    if $c_1 = False \vee c_2 = False \vee \varsigma_1 \prec^{PC}_{\varsigma} \varsigma_2 = \phi_\varsigma$
    then
      $@InterruptExecution$
    end if
  end for
end for
{Composition of protocol sequences}
Let $i = 1, j = 1$
Let $\xi = \{\langle \phi, \mathit{PRINIT}, \mathit{TINIT}, \phi, ., \phi\rangle\}$
while $i \leq n \wedge j \leq m$ do
  Let $\xi^i$ be the $i$-th element of $SEQ_1$
  Let $\xi^j$ be the $j$-th element of $SEQ_2$
  if $\xi \prec^C_\xi \xi^i \neq \phi_\xi$ then
    $\xi = \xi^i \prec^C_\xi \xi, i = i + 1$
  else if $\xi^i \prec^C_\xi \xi \neq \phi_\xi$ then
    $\xi = \xi \prec^C_\xi \xi^i, i = i + 1$
  end if
  if $\xi \prec^C_\xi \xi^j \neq \phi_\xi$ then
    $\xi = \xi^j \prec^C_\xi \xi, j = j + 1$
  else if $\xi^j \prec^C_\xi \xi \neq \phi_\xi$ then
    $\xi = \xi \prec^C_\xi \xi^j, j = j + 1$
  end if
end while
{Add remaining protocols}
while $i \leq n$ do
  $\xi = \xi \prec^C_\xi \xi^i, i = i + 1$
end while
while $j \leq m$ do
  $\xi = \xi \prec^C_\xi \xi^j, j = j + 1$
end while
```

sequences maintain their security properties requires applying the *PART_NONDESTR* predicate on each protocol pair and the verification of the independence of the participant chains by using the PC composition operator $\_ \prec_{\varsigma}^{PC} \_$. As shown in Algorithm 1, if one of these conditions is not satisfied, the execution is stopped, symbolized using the $@InterruptExecution$ keyword.

If the protocol properties are not destructive, the execution of the composition algorithm continues with the composition of protocol components. The final protocol is denoted by ξ, which, initially, contains a participant model with the effects *PRINIT* and types *TINIT*. These denote the initial knowledge for protocol participants, extracted from the context ctx, a unified context constructed from the contexts corresponding the the two sequences.

The composition process locates the position of each protocol in the final sequence by using the composition operator $\_ \prec_{\varsigma}^{C} \_$. If the result is ϕ_{ς}, the protocols can not be composed and another pair is selected. Finally, the remaining protocols are added to the sequence.

4 Experimental results

In order to validate the proposed method we generated several new composed protocols, based on existing ones. In order to verify if the new protocols accumulated the properties of the initial protocols, i.e. the composition is non-destructive, we applied the method proposed in this paper. However, such a verification is not enough for validating a method that must ensure the correctness of the resulted protocols, as shown by the large number of attacks discovered on protocols long after they have been published [3, 7].

Having these in mind, we turned to existing protocol verification tools. The purpose of the verification was to determine if new attacks became available on the composed protocols. One of the few tools allowing the verification of multi-protocol attacks is Scyther [4], which is the only tool currently available that also detects type-flaw attacks [19, 20], commonly found in multi-protocol environments.

We have applied our method on several pairs of security protocols defined in the library maintained by Clark and Jacob [21], for which there is also an online version available [22]. Through our experiments we composed protocol pairs such as CCITT X.509 v1 (i.e. X509v1) and CCITT X.509 v1c (i.e. X509v1c), BAN Concrete RPC (i.e. BAN-RPC) and Lowe-B (i.e. Lowe-BAN), Lowe-Denning-Sacco (i.e. L-D-S) and Kao-Chow v1 (i.e. K-Cv1), Lowe-Kerberos (i.e. Lowe-Kerb) and Neuman-Stubblebine (i.e. Neuman-S), Hwang-Neuman-Stubblebine (i.e. H-N-S) and Neuman-Stubblebine, Needham-Schroeder (i.e. Needh-S) and CCITT X.509 v1, Lowe-Needham-Schroeder (i.e. L-N-S) and ISO9798, Otway-Rees (i.e. Otway-R) and Lowe-BAN, Yahalom-Lowe (i.e. Y-L) and Kao-Chow v1, as shown in Table 1. The non-destructive property of the composed protocol was validated using the Scyther tool.

Table 1. Protocol composition results

Protocol 1	Protocol 2	PE (S1/S2)	PC (S1/S2)	Scyther
Lowe-B	ISO9798	N/Y	Y/Y	Y/Y
Lowe-B	X509v1	N/N	Y/Y	Y/Y
ISO9798	X509v1	Y/Y	Y/Y	Y/Y
ISO9798	X509v1c	Y/Y	Y/Y	Y/Y
X509v1	X509v1c	Y/Y	Y/Y	Y/Y
X509v1	X509v1c	Y/Y	Y/Y	Y/Y
BAN-RPC	Lowe-B	Y/Y	N/N	N/N
L-D-S	K-Cv1	Y/Y	N/N	N/N
K-Cv1	K-Cv2	Y/Y	Y/Y	Y/Y
L-D-S	Kerbv5	Y/Y	N/N	N/N
Lowe-Kerb	Neuman-S	Y/Y	N/N	N/N
H-N-S	Neuman-S	Y/Y	Y/Y	Y/Y
Needh-S	X509v1	Y/N	Y/Y	Y/Y
L-N-S	ISO9798	Y/N	Y/Y	Y/Y
Otway-R	Lowe-B	Y/N	Y/Y	Y/Y
SPLICE	Needh-S	Y/Y	Y/Y	Y/Y
TMN	Andr-RPC	Y/N	Y/Y	Y/Y
Y-L	K-Cv1	Y/Y	N/N	N/N

In Table 1, S1 indicates the protocol composition sequence P1-P2, while S2 indicates the sequence P2-P1. We used "Y" to indicate the successful composition of a sequence and "N" the failure of the composition process. By applying the proposed non-destructivity conditions we have discovered several new multi-protocol attacks. For example, in case of the protocol pair Yahalom-Lowe and Kao-Chow, we discovered a new attack that gives the intruder the possibility to replay valid messages from the Kao-Chow v1 (i.e. K-Cv1) protocol into the Yahalom-Lowe (i.e. Y-L) protocol. We have created a composed protocol and used the Scyther tool to verify it. The result was that 2 new attacks were possible. After correcting the problem by adding additional terms to the protocols messages in order for participants to be able to verify the validity of these messages, the Scyther tool did not detect any attacks, which was also confirmed by our method.

5 Related work

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

In [14], Guttman proposes a composition method based on predefined protocol primitives that are used to construct new, composed protocols. A similar approach is proposed by Choi [9], that additionally defines *bindings* in order to correctly connect different primitives. The previously mentioned approaches have not been designed to compose existing protocols, as the one proposed in this paper. We have only mentioned them here for completeness.

A. Datta et all [11, 12] propose the description of each composed protocol and of the final protocol as a set of equations. The composition process starts out from the initial protocol equations and tries to reach the properties modeled by the final equations. By doing so, they also prove the correctness of the final protocol. In case of this approach, the human factor plays an important role. As opposed to this, our approach can be fully automatized, eliminating the interference of the human factor.

The approach proposed by S. Andova et all [13] also uses equations written for each protocol and for each security property that must be satisfied by the final protocol. The composition process uses the human operator to construct the final properties from the initial equations and the Scyther [4] tool to automatically verify the correctness of the composed protocols. This approach is a semi-automatized one that uses the human operator to construct the final properties and an automatic verification tool for the verification of the correctness of the final protocol.

6 Conclusion and future work

We have developed a method for the composition of security protocols. The novelty of our approach is the fact that it provides a syntactical verification of the involved protocols, that makes it appropriate for on-line automated composition applications.

Our proposal makes use of an enriched protocol model that embodies protocol preconditions and effects. Messages exchanged by participants are modeled as sequences of nodes called participant chains. Based on this model we proposed conditions for the precondition-effect composition. This process involves determining if sufficient knowledge is provided by previous protocols and if instance-specific security properties are maintained even after the composition.

The protocol-chain composition process makes use of a canonical model that eliminates message component instances. This model reduces each component of the protocol model to its basic type. By doing so we are able to verify the instance-independent components of security protocols and detect multi-protocol attacks in a syntactical manner.

We have applied the proposed composition method on several pairs of well-known security protocols and have found new multi-protocol attacks. Our independence verification method has been validated using the security protocol verification tool Scyther, a state-space exploration method, by discovering the same multi-protocol attacks.

As future work, we intend to use the proposed composition method in the design process of new protocols for Web services. This would allow us to implement more complex protocols, such as TLS [23], currently used as a binary security protocol, using an XML message format that would enrich the properties of TLS with the ones specific to Web services such as extensibility or flexibility.

References

[1] C.J.F. Cremers, and S. Mauw, “Checking secrecy by means of partial order reduction”, In S. Leue and T. Systa, editors, Germany, september 7-12, 2003, revised selected papers LNCS, Vol. 3466, 2005.

[2] F.J.T. Fabrega, J.C. Herzog, and J.D. Guttman, “Strand spaces: Proving security protocols correct”, *Journal of Computer Security*, 1999, Vol. 7, pp. 191–230.

[3] C. Weidenbach, Towards an automatic analysis of security protocols, *Lecture Notes in Artificial Intelligence* Vol. 1632, 1999, pp. 378–382.

[4] C.J.F. Cremers, “Scyther”, Semantics and Verification of Security Protocols, Thesis, University Press Eindhoven, 2006.

[5] D. Dolev, and A.C. Yao, “On the security of public key protocols”, *IEEE Transactions on Information Theory*, Vol. 29, 1983, pp. 198–208.

[6] I. Cervesato, “The Dolev-Yao Intruder is the Most Powerful Attacker”, 16th Annual Symposium on Logic in Computer Science, LICS’01, IEEE Computer Society Press, Boston, MA, 2001.

[7] Gavin Lowe, “Some new attacks upon security protocols”, In Proceedings of the 9th Computer Security Foundations Workshop, IEEE Computer Society Press, 1996, pp. 162–169.

[8] C.J.F. Cremers, “Compositionality of Security Protocols: A Research Agenda”, Electr. Notes Theor. Comput. Sci., Vol. 142, 2006, pp. 99–110.

[9] H.J. Choi, “Security protocol design by composition”, Cambridge University, UK, Technical report Nr. 657, UCAM-CL-TR-657, ISSN 1476-2986, 2006.

[10] Ran Canetti, “Universally composable security: A new paradigm for cryptographic protocols”, 42nd FOCS, 2001, Revised version (2005), available at eprint.iacr.org/2000/067.

[11] A. Datta, A. Derek, J.C. Mitchell, and D. Pavlovic, “Secure Protocol Composition”, Proceedings of the 2003 ACM workshop on Formal methods in security engineering, 2003, pp. 11–23.

[12] A. Datta, A. Derek, J.C. Mitchell, and A. Roy, “Protocol Composition Logic (PCL)”, Electronic Notes in Theoretical Computer Science, Vol. 172, 1 April, 2007, pp. 311–358.

[13] S. Andova, C.J.F. Cremers, K. Gjosteen, S. Mauw, S. Mjolsnes, S. Radomirovic, “A framework for compositional verification of security protocols”, Special issue on computer security: Foundations and Automated Reasoning, Vol. 206(2-4), Elsevier, 2008, pp. 425–459.

[14] J.D. Guttman, “Security protocol design via authentication tests”, In Proceedings of the 15th IEEE Computer Security Foundations Workshop, IEEE CS Press, June, 2002.

[15] B. Genge, and I. Ignat, “Verifying the Independence of Security Protocols”, 3rd IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, 2007, pp. 155–163.

[16] C.J.F. Cremers, “Verification of multi-protocol attacks” Computer Science Report CSR 05-10, Eindhoven University of Technology, 2005.

[17] Organization for the Advancement of Structured Information Standards, “SAML V2.0 OASIS Standard Specification”, http://saml.xml.org/, 2007.

[18] Organization for the Advancement of Structured Information Standards, “OASIS Web Services Security (WSS)”, http://saml.xml.org/, 2006.

[19] J. Heather, G. Lowe, and S. Schneider, “How to Prevent Type Flaw Attacks on Security Protocols”, In the Proc. of the 13th Computer Security Foundations Workshop, IEEE Computer Society Press, July 2000.

[20] C. Meadows, “A Procedure for Verifying Security Against Type Confusion Attacks”, IEEE Computer Security Foundations Workshop (CSFW’03), 2003, pp. 62–70.

[21] J. Clark, J. Jacob, “A Survey of Authentication Protocol Literature: Version 1.0”, York University, 17 November 1997.

[22] Laboratoire Specification et Verification, “Security Protocol Open Repository”, http:// www.lsv.ens-cachan.fr/spore/, 2008.

[23] T. Dierks, and C. Allen, “The TLS Protocol Version 1.0”, Request for Comments: 2246, Network Working Group, January 1999.