Automation
Computers
Applied Mathematics
ISSN 1221–437X
Vol. 17 (2008) no. 2
pp. 169–178

Syntactic Sequential Composition of Security Protocols

Béla Genge and Iosif Ignat

Béla Genge: "Petru Maior" University of Târgu Mureş, Department of Electrical Engineering, N. Iorga Str., No. 1, (4300) Târgu Mureş, ROMANIA

bgenge@upm.ro

Iosif Ignat: Technical University of Cluj-Napoca, Department of Computer Science, Baritiu Str., No. 28, (3400) Cluj-Napoca, ROMANIA

Iosif.Ignat@cs.utcluj.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 construct an enriched protocol model for analyzing instance-related properties and a canonical model for analyzing message structure-related properties. The protocol model provides for each participant the preconditions needed to run the protocol, the effects resulted from running the protocol, the generated message components and the transmitted and received message sequences. The canonical model integrates participant knowledge in the model reducing each message component to its basic type. This allows us to conduct a syntactical analysis on the canonical model and to detect multi-protocol attacks that can be constructed by attackers in case of composed protocols. The proposed method ensures the sequential composition of protocols with the satisfaction of preconditions and non-destructive effects.

Key Words: Security protocols, sequential composition, syntactic model verification.

MSC 2000: 68M12, 68Q60

Received: November 11, 2008

1 Introduction

Security protocols are "communication protocols dedicated to achieving security goals" (C.J.F. Cremers and S. Mauw) [5] 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 [3, 4, 12]. The majority of these methods consider a Dolev-Yao-like intruder model [1, 2] to capture the actions available to a intruder which has complete control over the network. By analyzing each individual protocol in the presence of this penetrator model, the literature has reported numerous types of attacks [6, 4]. 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 [9].

Designing new protocols is a challenging task if we look at the number of attacks that have been discovered over the years [6] after the protocols have been published. However, in the last few years the use of protocol composition [7, 8, 9] has been successfully applied to create new protocols based on existing [10, 11] or predefined protocols [7].

By composing existing protocols we create new protocols that have a unified set of security properties. The composition process can combine protocol messages, known as parallel composition [10], or it can combine protocols without also combining protocol messages, known as sequential composition [11].

The composition method presented in this paper uses a syntactical analysis of two or more security protocols for determining existing security protocol attacks. In order to introduce this method, we first present a protocol model that includes information related to protocol preconditions, effects, participant knowledge, and the sequence of sent and received messages.

Based on this protocol model, we define a canonical model that allows a syntactical analysis of the modeled protocols by eliminating instance-based information through the use of message component *types*. By doing so, we syntactically model the knowledge of protocol participants, used to identify and verify message components.

In order to prove that our method provides a safe composition of security protocols we use the Dolev-Yao intruder model. We prove that if certain conditions are met, the intruder can not use any of its powers to construct new attacks based on messages extracted from other protocols. The method is also validated by verifying several composed protocols using the widely-adopted security protocol verification tool Scyther [12] which is one of the only verification tools that provides an intuitive multi-protocol analysis approaches by simply concatenating multiple security protocol specifications.

2 Protocol model

Protocol participants communicate by exchanging *terms* constructed from elements belonging to the following basic sets: P, denoting the set of role 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 [13, 14] 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*:

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

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 $\_^{-1} : \mathsf{K} \to \mathsf{K}$.

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

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

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 2.1 *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 2.2 *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 2.3 *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,* $part \in P$ *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 2.4 *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 not destructive.

In order to establish if the set of preconditions corresponding to a protocol can be satisfied based on the effects corresponding to another protocol we use the predicate $PART\_PREC : \mathsf{PR\_CC}^* \times \mathsf{PR\_CC}^*$. For two participant models, $\varsigma_1 = \langle prec_1, eff_1, type_1, gen_1, part_1, chain_1 \rangle$ and $\varsigma_2 = \langle prec_2, eff_2, type_2, gen_2, part_2, chain_2 \rangle$, the $PART\_PREC$ predicate is defined as

$$PART\_PREC(eff_1, prec_2) = \begin{cases} True, & \text{if } eff_1 \subseteq prec_2 \ , \\ False, & \text{otherwise} \ . \end{cases}$$

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

$$PART\_NONDESTR(eff_1, prec_2, eff_2) = \begin{cases} True, & \text{if } EF_1 \neq CON\_CONF \vee \\ & \quad \text{if } EF_1 = CON\_CONF \wedge t_1 = t_2 \text{ then } \exists EF_2(t_2) : EF_2 = CON\_CONF, \\ & \forall EF_1(t_1) \in eff_1 \wedge \forall PR_2(t_2) \in prec_2, \\ False, & \text{otherwise.} \end{cases}$$

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 $\varsigma_1 \prec_{\varsigma}^{PE} \varsigma_2 \neq \varsigma_2 \prec_{\varsigma}^{PE} \varsigma_1$.

The PE composition requirements of two participant models can easily be 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 $\xi_1 \prec_{\xi}^{PE} \xi_2 \neq \xi_2 \prec_{\xi}^{PE} \xi_1$.

3.2 Composition of protocol chains

The PC composition makes use of a canonical model that focuses on terms that can be verified by protocol participants. For each term from the protocol model, defined in the previous sections, 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 verify this we use an intruder model based on the Dolev-Yao [1, 2] model to capture the powers that can be used by an intruder. We prove that if certain conditions are met, the intruder can not use its powers to construct attacks on protocols based on messages extracted from other protocols.

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

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

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.

In the encryption process of the same plaintext, the use of two different keys, K_1 and K_2, will produce two different ciphertexts. This is also true for the decryption process, where the use of two different keys results in two different plaintexts. Because of this, we consider that the type of the encrypted terms after decryption will change too, according to the keys that are used. Thus we use an indexed key type k_i, such that $\mathsf{k}_i \neq \mathsf{k}_j$, where $i \neq j$, to distinguish between key types corresponding to different keys. In the definition of *BasicType*, the set of all typed keys is denoted by $\mathcal{K}$. We define the $\_^{-1} : \mathcal{K} \to \mathcal{K}$ function to map the canonical key pair corresponding to a given canonical key.

The *unknown* type u corresponds to terms that can not be validated because of limited role 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.

Based on the defined basic terms we can now proceed with the definition of *canonical terms* that makes use of the previously defined function names:

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

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

Definition 3.1 *A* canonical node *is any transmission or reception of a canonical term denoted as* $\langle \sigma, t \rangle$*, 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 3.2 *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 3.3 *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 3.4 *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.*

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 introduced in this sub-section. We expand these conditions to protocol models in the following definitions.

Definition 3.5 *Two protocol models* $\xi_1, \xi_2 \in$ MPROT *are* instance independent *if* $\forall\varsigma_1 \in \xi_1$*,* $\forall\varsigma_2 \in \xi_2$*, the predicates* $PART\_NONDESTR(eff_1, prec_2, eff_2)$ *and* $PART\_NONDESTR(eff_2, prec_1, eff_1)$ *hold, where* $prec_1$ *and* eff_1 *are the preconditions and effects corresponding to* ς_1 *and* $prec_2$ *and* eff_2 *are the preconditions and effects corresponding to* ς_2*.*

Definition 3.6 *Let* $csentEnc$: MPROT-C $\rightarrow \mathcal{T}^*$ *be a function that maps all canonical positive terms corresponding to a given participant model and let* $crecvEnc$: MPROT-C $\rightarrow \mathcal{T}^*$ *be a function that maps all canonical negative terms corresponding to a given participant model's processing chain. Two canonical protocol models* $\xi_1, \xi_2 \in$ MPROT-C *are* canonical independent *if* $\forall\varsigma_1 \in \xi_1$*,* $\forall\varsigma_2 \in \xi_2$ *and* $\forall t_1 \in csentEnc(\varsigma_1)$*,* $\forall t_2 \in crecvEnc(\varsigma_2)$*,* $\forall t_1' \in crecvEnc(\varsigma_1)$*,* $\forall t_2' \in csentEnc(\varsigma_2)$*, the predicates* $CONSTR(t_1, t_2)$*,* $CONSTR(t_2', t_1')$ *do not hold.*

In the above definition, the *CONSTR* predicate expresses the fact that a given canonical term can be constructed by instantiation from another canonical term. Informally, we consider that this construction is possible if canonical terms from the same position are equal or that the second term contains an undefined canonical term. The $CONSTR : \mathcal{T} \times \mathcal{T}$ predicate is defined as

$$CONSTR(t,t') = \begin{cases} True, & \text{if } t = t' \vee (t \in BasicType \wedge t' = \mathsf{u}) \vee \\ & \quad (t = \mathsf{u} \wedge t' \in BasicType), \\ CONSTR(t_1, t_1') \wedge & \text{if } (t = (t_1, t_2) \wedge t' = (t_1', t_2')) \vee \\ \quad CONSTR(t_2, t_2'), & \quad t = \{t_1\}_{f(t_2)} \wedge t' = \{t_1'\}_{f(t_2')} \wedge (t_2 = t_2' \vee t_2' = \mathsf{u}), \\ False, & \text{otherwise.} \end{cases}$$

We consider a regular intruder model for the protocol model and a canonical intruder model for the canonical protocol model. In both cases, the intruder powers are the same

and are modeled as the following operations: generate terms (**M**) using the sequence $\langle +t \rangle$; intercept terms (**I**) using the sequence $\langle -t \rangle$; repeat intercepted terms (**R**) using the sequence $\langle -t, +t, +t \rangle$; concatenate intercepted terms (**C**) using the sequence $\langle -t_1, -t_2, +(t_1, t_2) \rangle$; separate intercepted terms $\langle -(t_1, t_2), +t_1, +t_2 \rangle$; generate keys (**G**) using $\langle +k \rangle$ or $\langle +\mathsf{k} \rangle$; encrypt (**E**) using the sequence $\langle -k, -t, +\{t\}_{f(k)} \rangle$ or $\langle -\mathsf{k}, -t, +\{t\}_{f(\mathsf{k})} \rangle$; decrypt (**D**) using the sequence $\langle -k^{-1}, -\{t\}_{f(k)}, +t \rangle$ or $\langle -\mathsf{k}^{-1}, -\{t\}_{f(\mathsf{k})}, +t \rangle$.

Based on the intruder model we formulate the definition of protocol model independence as follows.

Definition 3.7 *Two protocol models* $\xi_1, \xi_2 \in \mathsf{MPROT}$ *are* independent *if there is no node sequence that can be constructed by the intruder that leads to the acceptance of a term in* ξ_1 *extracted from* ξ_2 *and vice-versa.*

Next, we prove, by using 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 based on terms extracted from other protocols.

Proposition 3.8 *Let* $\xi_1, \xi_2 \in \mathsf{MPROT}$ *be two protocol models and* $\xi_1', \xi_2' \in \mathsf{MPROT\text{-}C}$ *their corresponding canonical models. If* ξ_1,ξ_2 *are instance independent and* ξ_1', ξ_2' *are canonical independent, then* ξ_1 *and* ξ_2 *are independent.*

Proof. We show that if ξ_1, ξ_2 are instance independent, then the intruder can not construct valid cryptographic terms and if ξ_1', ξ_2' are canonical independent then the intruder can not replay valid terms from one model to another.

Let CONF_1 be the set of confidential terms from ξ_1 and CONF_2 the set of confidential terms from ξ_2. According to the instance-independence condition, if there exists a precondition predicate $PR(t)$ in ξ_2 for $t \in \mathsf{CONF}_1$ such that $PR = CON\_CONF$, then $t \in \mathsf{CONF}_2$. From this we have that the intruder can not obtain valid confidential terms from ξ_1 to construct cryptographic terms in ξ_2. This means that the intruder can not use its **G**, **E**, **D** powers to construct new cryptographic terms.

However, the intruder can still use its remaining powers to replay terms from one protocol to another. According to the canonical independence condition, canonical terms from one protocol are not accepted in the other protocol. By construction, on instantiation, the resulting terms will also not be accepted, resulting that the intruder can not use its remaining powers to construct attacks. ■

If two protocol models are independent, then their participant chains can be composed. We use the $\_ \prec_{\varsigma}^{PC} \_ : \mathsf{MPART} \times \mathsf{MPART} \rightarrow \mathsf{MPART}$ operator to denote the PC composition of protocol chains and the $\_ \prec_{\xi}^{PC} \_ : \mathsf{MPROT} \times \mathsf{MPROT} \rightarrow \mathsf{MPROT}$ operator to denote the PC composition of protocol models.

3.3 Composition of protocol models

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 $\_ \prec^{C} \_ : \mathsf{MPROT} \times \mathsf{MPROT} \rightarrow \mathsf{MPROT}$.

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 $\xi_1 \prec^{C} \xi_2 \prec^{C} \ldots \prec^{C} \xi_i$.

4 Experimental results

In order to validate our approach we used existing protocol verification tools. The purpose of the verification was to determine if new attacks become available when other protocols are also present and if these attacks are also discovered by our approach. One of the few tools allowing the verification of multi-protocol attacks is Scyther [12], which is the only tool currently available that also detects type-flaw attacks [15], commonly found in multi-protocol environments.

We have applied our method to several pairs of security protocols defined in the library maintained by Clark and Jacob [16], for which there is also an online version available [17]. Through our experiments we have verified the composition of protocol pairs such as Yahalom-Lowe and Kao-Chow, Lowe-Needham-Schroeder and ISO9798, Lowe-Denning-Sacco and Lowe-Wide-Mouthed-Frog, Andrew-Secure-RPC and CCITT X.509, Denning-Sacco and Otway-Reese.

By applying the independence conditions we have discovered several new multi-protocol attacks. For example, in case of the protocol pair Yahalom-Lowe and Kao-Chow, a new attack was discovered that gave the intruder the possibility to replay valid messages from the Kao-Chow protocol in the Yahalom-Lowe 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, the Scyther tool did not detect any attacks, which was also confirmed by our method.

5 Conclusion

We have developed a method for the sequential 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, constructed as a state-space exploration method, by discovering the same multi-protocol attacks.

References

[1] D. Dolev, A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29: 198–208, 1983.

[2] 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.

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

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

[5] C. Cremers, 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, Springer.

[6] 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.

[7] Hyun-Jin Choi. Security protocol design by composition. Cambridge University, UK, Technical report Nr. 657, UCAM-CL-TR-657, ISSN 1476-2986, 2006.

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

[9] Cas J. F. Cremers. Compositionality of Security Protocols: A Research Agenda. Electr. Notes Theor. Comput. Sci., 142, pp. 99–110, 2006.

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

[11] S. Andova, Cas J.F. Cremers, K. Gjosteen, S. Mauw, S. Mjolsnes, S. Radomirovic. A framework for compositional verification of security protocols", Elsevier, to appear, 2007.

[12] Cas Cremers, Scyther. Semantics and Verification of Security Protocols, Thesis, University Press Eindhoven, 2006.

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

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

[15] J. Heather, G. Lowe, 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.

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

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