AVISPA

www.avispa-project.org

IST-2001-39252

Automated Validation of Internet Security Protocols and Applications

Deliverable D4.1: Compositionality

Abstract

In this document, we report on three different approaches that we have investigated to tackle different aspects of compositionality: a general method for reasoning about contract-signing protocols using a specialised protocol logic, an algorithm for combining decision procedures for intruder constraints on disjoint signatures, and (Abstract) Secure Communication Channels as a means of modelling larger application protocols which make use of several sub-protocols and where one wishes to specify different intruder models for different parts of a protocol. Our experimental analyses show that these methods are very useful to further scale up the automated deduction techniques and tools that we have been developing in the AVISPA project.

Deliverable details

Deliverable version: *v1.0*
Person-months required: *8*
Date of delivery: *20.06.05*
Due on: *31.03.2005*
Classification: *public*
Total pages: *27*

Project details

Start date: *January 1st, 2003*
Duration: *30 months*
Project Coordinator: *Alessandro Armando*
Partners: *Università di Genova, INRIA Lorraine, ETH Zürich, Siemens AG*

Information Society Technologies

Project funded by the European Community under the *Information Society Technologies* Programme (1998-2002)

[This page has been intentionally left blank.]

Contents

1 Introduction

In order to improve the automated deduction techniques and tools that we have been developing in the AVISPA project, we have investigated different ways of exploiting the modular structure of complex protocols during analysis by using compositional reasoning techniques. This allows, for instance, properties of sub-protocols to be analysed in isolation and then combined to establish properties of the main protocol as a whole.

A number of systems of compositional inference rules have already been put forward to support "assume-guarantee" style of reasoning using different temporal logics [2, 42, 44], and much effort has also been devoted to devising formal approaches for reasoning about secure protocol composition, e.g. [16, 18, 26, 27, 40, 41]. In this deliverable, we describe three different approaches that we have investigated to tackle different aspects of compositionality.

First, as we describe in Section 2, in [9] we have developed a general method for reasoning about contract-signing protocols using a specialised protocol logic (based on the cord calculus [36]). We have applied this method to prove properties of the Asokan-Shoup-Waidner protocol [4, 5] and the Garay-Jacobson-MacKenzie protocol [38]. Our method offers several advantages over previous analysis techniques. First, it is compositional: the security guarantees are proved by combining the independent proofs for the three subprotocols of which each protocol is comprised. Second, the formal proofs are carried out in a "template" form, which gives us a reusable proof that may be instantiated for the ASW and GJM protocols, as well as for other protocols with the same arrangement of messages. Third, the proofs follow the design intuition. In particular, in proving game-theoretic properties like *fairness*, we demonstrated that the specific strategy that the protocol designer had in mind works, instead of showing that one exists. Finally, our results hold even when an unbounded number of sessions are executed in parallel.

Second, as we describe in Section 3, in [22] we have proposed an algorithm for combining decision procedures for intruder constraints on disjoint signatures. This algorithm allows for a modular treatment of algebraic operators in protocol analysis and we believe that it will contribute to a better understanding of complexity issues in the domain. Since constraint satisfiability is required only from the intruder sub-theories the approach should permit one to handle more complex protocols than with alternative techniques.

Third, as we describe in Section 4, the thesis [22] has proposed (Abstract) Secure Communication Channels (SCC) as a mean of modelling larger application protocols which make use of several sub-protocols and where one wishes to specify different intruder models for different parts of a protocol (i.e. channels). Moreover, SCCs enable for a top-down approach to protocol design and analysis, where protocol designers first specify their needs and can then incrementally instantiate a protocol with a concrete realisation of those needs.

As part of this research, we have introduced SCCs into HLPSL proposing syntax, semantics and tool implementation for different types of SCC, and then experimentally assessed the effectiveness of SCCs on three examples of real protocols which are difficult to model with the tools available today: Purpose Built Keys Framework (PBK [15]), the Asokan-Shoup-Waidner Contract Signing Protocol (ASW [4, 5]), and the Internet Open

Trading Protocol (IOTP [17]).

2 Compositional Protocol Analysis

Several methods have been used to analyse contract-signing protocols and either find errors or suggest their absence. For example, [52] uses the finite-state enumeration tool Murφ, [19] uses inductive arguments, etc.. However, previous studies only consider a bounded number of participants (usually an initiator, a responder, and a trusted third party) or involve arguments about an often confusing number of cases. In addition, with our method it is not necessary to consider interleaving of actions from different subprotocols, since properties of the entire protocol can be proved compositionally from independent proofs for the three subprotocols (called the *exchange*, *abort*, and *resolve* subprotocols in further sections).

We used the compositional protocol logic proposed in [37, 28], and originally intended for authentication protocols. Surprisingly, the logic proves appropriate to the task, requiring only minor modification. This logic is extended to prove correctness of protocol templates, which are abstract protocols containing function variables (a protocol function abstracted as a variable) for some of the operations used to construct messages. Correctness of a protocol template may be established under certain assumptions about these function variables. Then, a proof for an actual protocol is obtained by replacing the function variables with combinations of operations that satisfy the proof assumptions. We followed this method for a family of contract-signing protocols, establishing correctness of the Asokan-Shoup-Waidner and Garay-Jacobson-MacKenzie protocols by instantiation. Although the formal proofs reflect the intricacies associated with any formal logic, the proofs seem to be direct encodings of natural lines of argument. In addition to compositional reasoning about the combined properties of three independent subprotocols, the protocol logic does not require any explicit reasoning about the possible behaviour of any dishonest party, since the axioms and inference rules are sound for any hostile environment.

The structure of this section is a follows: we first present the studied protocol; then we try to give some ideas of the methodology used, as well as the analysed security properties; finally, we present the protocol templates.

2.1 Contract Signing Protocols

Asokan-Shoup-Waidner Protocol The protocol of Asokan, Shoup, and Waidner [4, 5], henceforth called simply the *ASW protocol*, consists of three interdependent subprotocols: *exchange*, *abort*, and *resolve*. The parties (an *originator* O and a *responder* R) generally start the exchange by following the *exchange* subprotocol. If both O and R are honest and there is no interference from the network, each obtains a valid contract upon completion of the *exchange* subprotocol. The originator O also has the option of requesting the trusted third party T to abort an exchange that O has initiated. To do so, O executes the *abort* subprotocol with T. Finally, both O and R may each request that T resolve an exchange

that has not been completed. After receiving the initial message of the *exchange* protocol, they may do so by executing the *resolve* subprotocol with T. In the Alice-Bob notation, this is:

Exchange protocol:

$O \to R: \quad me_1 = Sig_O\{V_O, V_R, T, text, Hash(N_O)\}$
$R \to O: \quad me_2 = Sig_R\{me_1, Hash(N_R)\}$
$O \to R: \quad me_3 = N_0$
$R \to O: \quad me_4 = N_R$

Two forms of contract:

standard : $\{me_1, N_O, me_2, N_R\}$
replacement : $Sig_T\{me_1, me_2\}$

Abort protocol:

$O \to T: \quad Sig_O\{abort, me_1\}$
$T \to O: \quad Sig_T\{me_1, me_2\}$ if resolved alr.
or $T \to O: \quad Sig_T\{aborted, ma_1\}$ otherwise

Resolve protocol:

$R \to T: \quad me_1, me_2$
$T \to R: \quad Sig_T\{me_1, me_2\}$ if resolved alr.
or $T \to R: \quad Sig_T\{aborted, ma_1\}$ otherwise

Garay-Jakobsson-MacKenzie Protocol The protocol of Garay, Jakobsson, and MacKenzie, henceforth simply called the *GJM protocol*, is closely related to the ASW protocol in that both protocols involve a 4-step *exchange* subprotocol and similar *abort* and *resolve* subprotocols. Even though the two protocols have similar structure, the actual contents of the messages differ. The GJM protocol relies on the cryptographic primitive called *private contract signature* (PCS). Let O be a signing party, R a receiver, T a trusted third party, and m a *PCS* generated by O for R. Then the main properties of PCS are as follows: (a) m can be verified by R like a conventional signature; (b) m can be feasibly computed by either O, or R, but nobody else; (c) m can be converted into a conventional signature by either O, or T, but nobody else, including R. Converted signatures are universally verifiable by anybody in possession of the correct signature verification key. Using these properties, GJM consists in O and R first exchanging PCS (i.e. $PCS_O(text, R, T)$ and $PCS_R(text, O, T)$), and second exchanging true signatures (i.e. $Sig_O(text)$ and $Sig_R(text)$) as messages me_1 to me_4. The abort and resolve protocols are modified accordingly, as well as the standard and replacement contracts.

2.2 Methodology

We used the *cord calculus* to model the protocols and their executions (see [36] for a detailed description of *cords*). The basic protocol logic and proof system used in this work are developed in [36, 28, 29, 31], with [32] providing a relatively succinct presentation of the most recent form. The formulae of the logic are given by the following grammar:

$$\begin{array}{rcl} \mathsf{a} & ::= & \mathsf{Send}(P,m) \mid \mathsf{Receive}(P,m) \mid \mathsf{New}(P,t) \mid \mathsf{Decrypt}(P,t) \mid \mathsf{Verify}(P,t) \mid \mathsf{Start}(P) \\ \phi & ::= & \mathsf{a} \mid \mathsf{Has}(P,t) \mid \mathsf{Computes}(P,t) \mid \mathsf{Fresh}(P,t) \mid \mathsf{Honest}(N) \mid \phi \wedge \phi \mid \neg\phi \mid \exists x.\phi \mid \mathsf{a} \leq \mathsf{a} \\ \Psi & ::= & \phi\ [\rho]_X\ \phi \qquad \text{(modal formula) with } \rho \text{ a list of actions a for participant } X \end{array}$$

where ρ may be any role, written using the notation of the cord calculus. Here, t and P denote a term and a thread, respectively. A *thread* is a principal executing an instance

of a role. Most protocol proofs use formulae of the form $\theta[P]_X\phi$, which means that after actions P are executed in thread X, starting from a state where formula θ is true, formula ϕ is true about the resulting state of X. Informally, the predicate interpretations are quite intuitive: $\mathsf{Has}(X, x)$ means principal $\hat{X}$ knows or can create the information x in the thread X; $\mathsf{Send}(X, m)$ means principal $\hat{X}$ sends message m in the thread X; $\mathsf{Honest}(\hat{X})$ means $\hat{X}$ follows exactly the protocol specification; and so on.

Runs of the ASW protocol There are three possible execution scenarios for the initiator in the ASW protocol. The initiator can complete the exchange subprotocol; complete the resolve subprotocol after sending the third message of the exchange subprotocol; or complete the abort protocol after sending the first message of the exchange subprotocol. The design intent was that each of these three combinations should result in the initiator obtaining a valid contract whenever the responder already has one (see [5] for a more detailed discussion). Using the protocol logic, we are able to analyse the components of the ASW protocol independently, and combine the proofs using the composition theorems presented in [28, 32].

Formulae of the logic and sequential composition Most logical statements that we use are of the form $\Gamma \vdash \phi[\mathbf{P}]_A\theta$. The intuitive reading of such statement is: "Given that the set of assumptions Γ holds in every state, and a thread A has finished executing the program $\mathbf{P}$ starting in a state where ϕ holds, then in the resulting state θ holds." In this work, Γ typically contains a set of assumptions about the behaviour of the trusted third party T which can be later discharged by analysing T's program. For example, one of the assumptions we use is that T never sends a replacement contract if it has issued the abort token in the past.

In order to build security proofs, we can combine statements about different subprotocols using *sequential composition* of the roles. We use a variant of the theorem from [32]. It states that for sets of assumptions Γ_1 and Γ_2, if $\Gamma_1 \vdash \phi[\mathbf{P}]_X\theta$ and $\Gamma_2 \vdash \theta[\mathbf{Q}]_X\psi$ then $\Gamma_1 \cup \Gamma_2 \vdash \phi[\mathbf{P}; \mathbf{Q}]_X\psi$. This composition theorem is the key to build security proofs on roles by proving individual properties on each role's actions. (Note that parallel composition of roles is implicit in the protocol logic, and does not require any specific theorem.)

2.3 Protocol properties

We now list the properties that must be validated by an optimistic contract signing protocol like ASW. Complete details on these properties, as well as their precise formalisation and proof in our logic, can be found in [9]. We give a formalisation of one of the fairness properties as an example.

2.3.1 Fairness

Informally, in this context, fairness means that after the protocol has been successfully completed, either both parties have a signed contract or neither does. In our model,

fairness is expressed using a set of logical formulae. As described previously, we look separately at the three possible scenarios for the initiator to complete the protocol. We consider the initiator's point of view; a similar property can be shown for the responder.

The initiator completes the exchange subprotocol This is the optimistic part of the protocol. Formula ϕ_0 states that the initiator A has a valid contract after the successful execution of the exchange protocol. In our logic, this is:

$$\phi_0 \equiv \mathsf{Start}(A)\ \Big[\mathbf{ExchangeInit}(\hat{A},\hat{B},\hat{T},text)\Big]_A\ \mathsf{Has}(A, s(\hat{A},\hat{B},\hat{T},text,x,y))$$

with **ExchangeInit**(...) the initiator's role written in cord calculus, and $s(...)$ the standard contract. By using the logic axioms and rules, we show that $\vdash \phi_0$. Therefore, in this scenario fairness holds without any assumptions about the behaviour of the trusted third party or the responder in the protocol.

The initiator runs the abort protocol If A started the protocol as the initiator but did not complete it, we show that whenever some other party has a valid contract, then A will get the replacement contract if it executes the abort subprotocol after sending the first message. This part of the analysis is done using the compositional proof method.

The initiator runs the resolve protocol Finally, we show that if A has received the second message of the protocol, then it can obtain a valid contract by executing the resolve subprotocol, provided that it created the nonce x itself and did not send the abort message corresponding to that nonce in the past.

2.3.2 Accountability

Accountability means that if one of the parties gets cheated as a result of $\hat{T}$'s misbehaviour, then it will be able to hold $\hat{T}$ accountable. More precisely, at the end of every run where an agent gets cheated, its trace together with a contract of the other party should provide non-repudiable evidence that $\hat{T}$ misbehaved. This property is expressed and proved in our logic.

2.3.3 Abuse-Freeness

Abuse-freeness means that no party can ever prove to the third party that it has the power to both enforce and cancel the contract. More precisely, a protocol is abuse-free for the initiator if in every state where the responder has publicly verifiable information that the initiator is bound to the contract, it has to be that the responder is also bound to the contract. (See [9] for proof details.)

2.4 Template for Optimistic Contract Signing Protocols

Both the ASW and GJM protocols consist of three interdependent subprotocols: *exchange*, *abort*, and *resolve*. The structure of these two protocols suggests a general pattern for two-party optimistic contract-signing protocols. Specifically, the *exchange* subprotocol proceeds in two stages. In the first stage, the two parties commit to the contract and in the second they open their commitment, in effect, ensuring that they are bound to the contract. Given this observation, it seems natural to ask if we could provide a unified representation and proof for these two protocols and their properties. We answered this question in the affirmative. The technical machinery used toward this end is presented in [30], and the proof details in [9].

A protocol template is a protocol that uses function variables replacing some messages or message parts, like the exchanged messages me_1 to me_4, or the contracts, in the ASW's description. A template is instantiated to a concrete protocol by fixing specific values of these functions. The structure of a security proof for a template protocol in our formalism consists of three steps: first, define a (small) set of axioms (hypothesis) on the behaviour of the functions in the template; second, prove the security of the template's properties assuming these axioms are satisfied; last, prove that each template instantiation we are interested in satisfy these axioms. This method was successfully applied to the ASW and GJM protocols. Let us explain how. In the particular case of these protocols, we had to : first, identify the differences of functionalities between the ASW and GJM protocols and abstract them. This gives us a (small) set of axioms representing some elementary properties of these protocols that cannot be proved in common. In the PCL logic, these are called hypothesis. Second, any other property, and in particular main security properties of ASW and GJM, can be proved for these protocols together with abstracted functionalities. This is the template protocol, and a proof for it consists of a combination of the PCL axioms and rules, extended with the previous template hypothesis. Third, we need to prove that both ASW and GJM satisfy the previous template hypothesis. This must naturally be proved separately for both ASW and GJM (the two template instances), since it represents the true differences between these protocols. All together, these tree steps prove that both ASW and GJM are secure. Naturally, the great advantage of templates is that each template instantiation (i.e. ASW or GJM here, but there could be more) requires only limited proofs of the template hypothesis, while other proofs are done in common. This also shows that protocol templates provide a useful method for formally characterising design concepts.

3 Combining Intruder Theories

Most of the decision procedures for symbolic analysis of protocols are limited to a fixed set of algebraic operators associated with a fixed intruder theory. Examples of such sets of operators comprise *exclusive-or* XOR, multiplication/exponentiation, abstract encryption/decryption. In the following, we give an algorithm for combining decision procedures

for arbitrary intruder theories with disjoint sets of operators, provided that solvability of ordered intruder constraints, a slight generalisation of intruder constraints, can be decided in each theory (a more detailed description of this algorithm can be found in [22]). This is the case for most of the intruder theories for which a decision procedure has been given. In particular, our result allows us to decide trace-based security properties of protocols that employ any combination of the above mentioned operators with a bounded number of sessions.

3.1 Introduction

Recently, many procedures have been proposed to decide insecurity of security protocols in the Dolev-Yao model with respect to a finite number of protocol sessions [3, 12, 50]. Among the different approaches the symbolic ones [48, 23, 11] are based on reducing the problem to constraint solving in a term algebra. This reduction has proved to be quite effective on standard benchmarks and also allowed for the discovery of new flaws on several protocols [11].

However, while most formal analysis of security protocols abstracts from low-level properties, i.e., certain algebraic properties of encryption, such as the multiplicative properties of RSA or the properties induced by chaining methods for block ciphers, many real attacks and protocol weaknesses rely on these properties. For attacks exploiting the properties of XOR in the context of mobile communications see [14]. Also the specification of *Just Fast Keying* protocol (an alternative to IKE) in [1] employs a set constructor that is idempotent and commutative and a Diffie-Hellman exponentiation operator with the property $(g^y)^z = (g^z)^y$.

We here present a general procedure for deciding security of protocols in presence of algebraic properties. This procedure relies on the combination of constraint solving algorithm for disjoint intruder theories, provided that solvability of ordered intruder constraints, a slight generalisation of intruder constraints, can be decided in each theory. Such combination algorithm already exists for solving E-unification problems [51, 8]. We have extended it in order to solve intruder constraints on disjoint signatures. This extension is non trivial since intruder deduction rules allow one to build *contexts* above terms and therefore add some second-order features to the *standard* first-order E-unification problem.

Our approach is more modular than the previous ones and it allows us to decide interesting intruder theories that could not be considered before by reducing them to simpler and independent theories. For instance it allows one to combine the exponentiation with Abelian group theory of [47] with the XOR theory of [21].

Related work. Several protocol decision procedures have recently been designed for handling algebraic properties in the Dolev-Yao model [45, 13, 24, 21]. These works have been concerned by fixed equational theories corresponding to a fixed intruder power. A couple of works only have tried to derive generic decidability results for *classes* of intruder theories. For instance, in [33] Delaune and Jacquemard consider the class of *public collaps-*

ing theories. These theories have to be presented by rewrite systems where the right-hand side of every rule is a ground term or a variable, which is a strong restriction.

3.2 Motivation

3.2.1 Combination of algebraic operators

We consider in this section the Needham–Schroeder Public-Key protocol. This well-known protocol is described in the Alice and Bob notation by the following sequence of messages, where the comma denotes a pairing of messages and $\{M\}K_a$ denotes the encryption by the public key K_a of A.

$$\begin{aligned} &A \to B : \{N_a, A\}K_b \\ &B \to A : \{N_a, N_b\}K_a \\ &A \to B : \{N_b\}K_b \end{aligned}$$

Assume now that the encryption algorithm follows the El-Gamal encryption scheme (see http://en.wikipedia.org/wiki/ElGamal for more information on this scheme). The public key of A is defined by three publicly-available parameters: a modulus p_a, a base g_a and the proper public key $(g_a)^a \bmod p_a$. The private key of A is a. Denoting $\exp_p$ the exponentiation modulo p, and with new nonces k_1, k_2 and k_3, we can rewrite the protocol as:

$$\begin{aligned} &A \to B : \exp_{p_b}(g_b, k_1), (N_a, A) \oplus \exp_{p_b}(\exp_{p_b}(g_b, b), k_1) \\ &B \to A : \exp_{p_a}(g_a, k_2), (N_a, N_b) \oplus \exp_{p_a}(\exp_{p_a}(g_a, a), k_2) \\ &A \to B : \exp_{p_b}(g_b, k_3), (N_b) \oplus \exp_{p_b}(\exp_{p_b}(g_b, b), k_3) \end{aligned}$$

In this simple example we would like to model the group properties of XOR ($\oplus$), the associativity of exponential ($(x^y)^z = x^{y \times z}$), and the group properties of the exponents. Several works have already been carried out to take into account these algebraic properties for detecting attacks on a bounded number of sessions. However, none of these works handles the analysis of protocols combining several algebraic operators like the example above. The algorithm given here will allow one to decide the trace-based security properties of such protocols.

3.2.2 Examples of intruder theories

A convenient way to specify intruder theories in the context of security protocols is by giving a set L of *deduction rules* describing how the intruder can construct new messages from the ones he already knows and a set of *equational laws* $\mathcal{E}$ verified by the functions employed in messages. We give here two examples of intruder theories; some other theories are given in [22].

Abelian group theory. This intruder may treat messages as elements of an Abelian group. We assume here there is only one such group and that the composition law is $\cdot \times \cdot$, the inverse law is $\mathrm{i}(\cdot)$ and the neutral element is denoted by 1.

$$L_\times \left\{ \begin{array}{rcl} & \to & 1 \\ x & \to & \mathrm{i}(x) \\ x, y & \to & x \times y \end{array} \right. \qquad \mathcal{E}_\times \left\{ \begin{array}{rcl} (x \times y) \times z & = & x \times (y \times z) \\ x \times y & = & y \times x \\ 1 \times x & = & x \\ x \times \mathrm{i}(x) & = & 1 \end{array} \right.$$

Dolev Yao with explicit destructors. The intruder is given with a pairing operator and projections to retrieve the components of a pair. There are symmetric encryption ($\mathrm{se}(\_,\_)$) and decryption ($\mathrm{sd}(\_,\_)$) operators. For conciseness we omit the public-key encryption specification.

$$L_{DY} \left\{ \begin{array}{rcl} x, y & \to & \langle x, y \rangle \\ x & \to & \pi_1(x) \\ x & \to & \pi_2(x) \\ x, y & \to & \mathrm{se}(x, y) \\ x, y & \to & \mathrm{sd}(x, y) \end{array} \right. \qquad \mathcal{E}_{DY} \left\{ \begin{array}{rcl} \pi_1(\langle x, y \rangle) & = & x \\ \pi_2(\langle x, y \rangle) & = & y \\ \mathrm{sd}(\mathrm{se}(x, y), y) & = & x \end{array} \right.$$

3.3 Terms and subterms

We consider an infinite set of free constants C and an infinite set of variables $\mathcal{X}$. For all signatures $\mathcal{G}$ (i.e., a set of function symbols with arities), we denote by $\mathrm{T}(\mathcal{G})$ the set of terms over $\mathcal{G} \cup \mathrm{C}$, and we denote by $\mathrm{T}(\mathcal{G}, \mathcal{X})$ the set of terms over $\mathcal{G} \cup \mathrm{C} \cup \mathcal{X}$. The former is called the *set of ground terms over* $\mathcal{G}$, while the latter is called the *set of terms over* $\mathcal{G}$. Variables are denoted by x, y, v, terms are denoted by s, t, u, and finite sets of terms are written $E, F, ...$, and decorations thereof, respectively.

A *constant* is either a free constant or a function symbol of arity 0. Given a term t, we denote by $\mathrm{Var}(t)$ the set of variables occurring in t and by $\mathrm{Cons}(t)$ the set of constants occurring in t. We denote by $\mathrm{Atoms}(t)$ the set $\mathrm{Var}(t) \cup \mathrm{Cons}(t)$. We denote by $\mathcal{A}$ the set of all constants and variables. A substitution σ is an involutive mapping from $\mathcal{X}$ to $\mathrm{T}(\mathcal{G}, \mathcal{X})$ such that $\mathrm{Supp}(\sigma) = \{x \mid \sigma(x) \neq x\}$, the *support* of σ, is a finite set. The application of a substitution σ to a term t (respectively, a set of terms E) is denoted $t\sigma$ (respectively, $E\sigma$) and is equal to the term t (respectively, E) where all variables x have been replaced by the term $x\sigma$. A substitution σ is *ground* with respect to $\mathcal{G}$ if the image of $\mathrm{Supp}(\sigma)$ is included in $\mathrm{T}(\mathcal{G})$.

Here, we consider 2 disjoint signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, and 2 consistent equational theories $\mathcal{E}_1$ and $\mathcal{E}_2$ on $\mathcal{F}_1$ and $\mathcal{F}_2$, resp. We denote by $\mathcal{F}$ the union of the signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, $\mathcal{E}$ the union of the theories $\mathcal{E}_1$ and $\mathcal{E}_2$. A term t in $\mathrm{T}(\mathcal{F}_1, \mathcal{X})$ (resp. in $\mathrm{T}(\mathcal{F}_2, \mathcal{X})$) is called a *pure* 1-term (resp. a pure 2-term).

The *set* $\mathrm{Sub}_{\mathrm{syn}}(t)$ *of syntactic subterms* of a term t is defined recursively as follows. If t is a variable or a constant then $\mathrm{Sub}_{\mathrm{syn}}(t) = \{t\}$. If $t = f(t_1, \ldots, t_n)$ then $\mathrm{Sub}_{\mathrm{syn}}(t) = \{t\} \cup \bigcup_{i=1}^{n} \mathrm{Sub}_{\mathrm{syn}}(t_i)$. The *positions* in a term t are defined recursively as usual (i.e., as sequences of integers), ϵ being the empty sequence. We denote by $t_{|p}$ the syntactic subterm of t at position p. We denote by $t[p \leftarrow s]$ the term obtained by replacing in t the syntactic subterm $t_{|p}$ by s. We denote by $\mathrm{Sign}(\cdot)$ the function that associates to each term $t \notin \mathrm{C} \cup \mathcal{X}$

the signature ($\mathcal{F}_1$, or $\mathcal{F}_2$) of its symbol at position ϵ. For $t \in \mathrm{C} \cup \mathcal{X}$, we define $\mathrm{Sign}(t) = \bot$, with $\bot$ a new symbol. The term s is *alien* to u if $\mathrm{Sign}(s) \neq \mathrm{Sign}(u)$.

We define the set of *factors* of a term t, and denote $\mathrm{Factors}(t)$, the set of maximal syntactic subterms of t that are either alien to t or atoms and different from t. In particular $\mathrm{Factors}(t) = \emptyset$ for $t \in \mathcal{A}$.

We now define the notion of *subterm values*. Given a term t, the set of its subterm values is denoted by $\mathrm{Sub}(t)$ and is defined recursively by: $\mathrm{Sub}(t) = \{t\} \cup \bigcup_{u \in \mathrm{Factors}(t)} \mathrm{Sub}(u)$. For a set of terms E, $\mathrm{Sub}(E)$ is defined as the union of the subterms values of the elements of E.

As an example, consider $\mathcal{F}_1 = \{\oplus, 0\}$ and $\mathcal{F}_2 = \{f\}$ where f has arity 1. Then $\mathrm{Sub}(a \oplus (b \oplus 0)) = \{a \oplus (b \oplus 0), a, b, 0\}$. On the other hand $\mathrm{Sub}(f(b \oplus c)) = \{f(b \oplus c), b \oplus c, b, c\}$. This shows the difference with the notion of *syntactic subterms*. In the following, unless otherwise indicated, *the notion of subterm will refer to subterm values.*

3.3.1 Congruences and ordered rewriting.

We will introduce the notion of *ordered rewriting* [34], which is a useful technique that has been utilised (e.g. [8]) for proving the correctness of combination of unification algorithms.

Let $<$ be a simplification ordering on $\mathrm{T}(\mathcal{G})$ assumed to be total on $\mathrm{T}(\mathcal{G})$ and such that the minimum for $<$ is a constant $\mathrm{c}_{\min} \in \mathrm{C}$ (by definition, $<$ satisfies for all $s, t, u \in \mathrm{T}(\mathcal{G})$ $s < t[s]$ and $s < u$ implies $t[s] < t[u]$). Given a possibly infinite set of equations $\mathcal{O}$ on the signature $\mathrm{T}(\mathcal{G})$ we define the ordered rewriting relation $\rightarrow_{\mathcal{O}}$ by $s \rightarrow_{\mathcal{O}} s'$ iff there exists a position p in s, an equation $l = r$ in $\mathcal{O}$ and a substitution τ such that $s = s[p \leftarrow l\tau]$, $s' = s[p \leftarrow r\tau]$, and $l\tau > r\tau$.

It has been shown (see [34]) that by applying the *unfailing completion procedure* [43] to a set of equations $\mathcal{H}$ we can derive a (possibly infinite) set of equations $\mathcal{O}$ such that:

1. the congruence relations $=_{\mathcal{O}}$ and $=_{\mathcal{H}}$ are equal on $\mathrm{T}(\mathcal{F})$.
2. $\rightarrow_{\mathcal{O}}$ is convergent (i.e., terminating and confluent) on $\mathrm{T}(\mathcal{F})$.

We will say that $\mathcal{O}$ is an *o-completion* of $\mathcal{H}$. The relation $\rightarrow_{\mathcal{O}}$ being convergent on ground terms we can define $(t)\downarrow_{\mathcal{O}}$ as the unique normal form of the ground term t for $\rightarrow_{\mathcal{O}}$. Given a ground substitution σ, we denote by $(\sigma)\downarrow_{\mathcal{O}}$ the substitution with the same support such that for all variables $x \in \mathrm{Supp}(\sigma)$ we have $x(\sigma)\downarrow_{\mathcal{O}} = (x\sigma)\downarrow_{\mathcal{O}}$. A substitution σ is *normal* if $\sigma = (\sigma)\downarrow_{\mathcal{O}}$. We will denote by R an o-completion of $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. We denote by $\mathrm{C}_{\mathrm{spe}}$ the set containing the constants in $\mathcal{F}$ and $\mathrm{c}_{\min}$.

3.4 Protocols, intruders and constraint systems

Security of a given protocol is assessed with respect to a class of environments in which the protocol is executed. Dolev and Yao have described the environment not in terms of possible attacks on the protocol but by the deduction an intruder attacking a protocol execution is able to perform.

In Subsection 3.4.1, we define an extension of the Dolev-Yao model to arbitrary operators for modelling the possible deductions of the intruder. In Subsection 3.4.2 we define the protocol semantics for an execution within an hostile environment controlled by the intruder and in Subsection 3.4.3 we describe how we represent this execution by a constraint system.

3.4.1 Intruder systems

We will model messages as ground terms and intruder deduction rules as rewrite rules on sets of messages representing the knowledge of an intruder. An intruder derives new messages from a given (finite) set of messages by applying intruder rules. Since we assume some equational axioms $\mathcal{H}$ are verified by functions symbols in the signature, all these derivations have to be considered *modulo* the equational congruence $=_{\mathcal{H}}$ generated by these axioms.

An intruder deduction rule in our setting is specified by a term t in some signature $\mathcal{G}$. Given values for the variables of t, the intruder is able to generate the corresponding instance of t.

Definition 1. *An* intruder system $\mathcal{I}$ *is given by a triple* $\langle \mathcal{G}, T, \mathcal{H} \rangle$*, where* $\mathcal{G}$ *is a signature,* $T \subseteq \mathrm{T}(\mathcal{G}, \mathcal{X})$*, and* $\mathcal{H}$ *is a set of axioms between terms in* $\mathrm{T}(\mathcal{G}, \mathcal{X})$*. To each* $t \in T$ *we associate a* deduction rule $\mathrm{L}^t : \mathrm{Var}(t) \to t$ *and* $\mathrm{L}^{t,\mathrm{g}}$ *denotes the set of ground instances of the rule* L^t*:*

$$\mathrm{L}^{t,\mathrm{g}} = \{l \to r \mid \text{there is a ground substitution } \sigma \text{ on } \mathcal{G} \text{ such that } l = \mathrm{Var}(t)\sigma \text{ and } r =_{\mathcal{H}} t\sigma\}$$

The set of rules $\mathrm{L}_{\mathcal{I}}$ *is defined as the union of the sets* $\mathrm{L}^{t,\mathrm{g}}$ *for all* $t \in T$*.*

Each rule $l \to r$ in $\mathrm{L}_{\mathcal{I}}$ defines an intruder deduction relation $\to_{l \to r}$ between finite sets of terms. Given two finite sets of terms E and F, we define $E \to_{l \to r} F$ if and only if $l \subseteq E$ and $F = E \cup \{r\}$. We denote by $\to_{\mathcal{I}}$ the union of the relations $\to_{l \to r}$ for all $l \to r$ in $L_{\mathcal{I}}$ and by $\to_{\mathcal{I}}^*$ the transitive closure of $\to_{\mathcal{I}}$. We simply denote by $\to$ the relation $\to_{\mathcal{I}}$ when there is no ambiguity about $\mathcal{I}$. For instance, we can define $\mathcal{I}_\times = \langle \{\times, \mathrm{i}, 1\}, \{x \times y, \mathrm{i}(x), 1\}, \mathcal{E}_\times \rangle$ and we have $a, b, c \to_{\mathcal{I}_\times} a, b, c, c \times a$ by applying the rule $c, a \to c \times a \in \mathrm{L}^{x \times y, \mathrm{g}}$.

A *derivation* D of length n, with $n \geq 0$, is a sequence of steps of the form $E_0 \to_{\mathcal{I}} E_0 \cup \{t_1\} \to_{\mathcal{I}} \cdots \to_{\mathcal{I}} E_n$ with finite sets of ground terms $E_0, \ldots E_n$, and ground terms $t_1, \ldots, t_n$, such that $E_i = E_{i-1} \cup \{t_i\}$ for every $i \in \{1, \ldots, n\}$. The term t_n is called the *goal* of the derivation.

We define $\overline{E}^{\mathcal{I}}$ to be equal to the set $\{t \mid \text{there is an } F \text{ such that } E \to_{\mathcal{I}}^* F \text{ and } t \in F\}$, i.e., the set of terms that can be derived from E. If there is no ambiguity on the deduction system $\mathcal{I}$ we write $\overline{E}$ instead of $\overline{E}^{\mathcal{I}}$.

Let $\mathcal{O}$ be an o-completion of $\mathcal{H}$. We will assume from now that all the deduction rules generate terms that are normalised by $\to_{\mathcal{O}}$ and the goal and the initial set are in normal form for $\to_{\mathcal{O}}$. It can be shown, as we do in [22], that this is not restrictive for our main decidability result.

Given a set of terms $T \subseteq \mathrm{T}(\mathcal{G}, \mathcal{X})$, we define the set of terms $\langle T \rangle$ to be the minimal set such that $T \subseteq \langle T \rangle$ and for all $t \in \langle T \rangle$ and for all substitutions σ with image included in $\langle T \rangle$, we have $t\sigma \in \langle T \rangle$. Hence, terms in $\langle T \rangle$ are built by composing terms in T iteratively. We can prove easily that the intruder systems $\mathcal{I} = \langle \mathcal{G}, T, \mathcal{H} \rangle$ and $\mathcal{J} = \langle \mathcal{G}, \langle T \rangle, \mathcal{H} \rangle$ define the same sets of derivable terms, i.e., for all E we have $\overline{E}^{\mathcal{I}} = \overline{E}^{\mathcal{J}}$.

We want to consider the union of the two intruder systems $\mathcal{I}_1 = \langle \mathcal{F}_1, T_1, \mathcal{E}_1 \rangle$ and $\mathcal{I}_2 = \langle \mathcal{F}_2, T_2, \mathcal{E}_2 \rangle$. In particular, we are interested in the derivations obtained by using $\rightarrow_{\mathcal{I}_1} \cup \rightarrow_{\mathcal{I}_2}$. It can be noticed that $\langle T_1 \cup T_2 \rangle = \langle \langle T_1 \rangle \cup \langle T_2 \rangle \rangle$. Hence, by the remarks above, the terms derivable using $\langle T_1 \cup T_2 \rangle$ or $\langle T_1 \rangle \cup \langle T_2 \rangle$ are the same. For technical reasons, it will be more convenient to use $\langle T_1 \rangle \cup \langle T_2 \rangle$ for defining the union of two intruder systems:

Definition 2. *The* union of the two intruder systems $\langle \mathcal{F}_1, T_1, \mathcal{E}_1 \rangle$ *and* $\langle \mathcal{F}_2, T_2, \mathcal{E}_2 \rangle$ *is the intruder system* $\mathcal{U} = \langle \mathcal{F}, \langle T_1 \rangle \cup \langle T_2 \rangle, \mathcal{E} \rangle$.

3.4.2 Protocol analysis

In this subsection, we describe how protocols are modelled. In the following, we only model a single session of the protocol since it is well-known how to reduce several sessions to this case. Our semantics follows the one of [33].

In the Dolev-Yao model, the intruder has complete control over the communication medium. We model this by considering that the intruder *is* the network. Messages sent by honest agents are sent directly to the intruder and messages received by the honest agents are always sent by the intruder. From the point of view of the intruder, a finite execution of a protocol is the interleaving of a finite sequence of messages he has to send and a finite sequence of messages he receives (and add to his knowledge).

We also assume that the interaction of the intruder with an agent is an atomic step. The intruder sends a message m to an honest agent, this agent tests the validity of this message and responds to it. Alternatively, an agent may initiate an execution and in this case we assume it reacts to a dummy message $\mathrm{c}_{\min}$ sent by the intruder.

A *step* is a triplet $(\textsc{recv}(x); \textsc{send}(s); \textsc{cond}(e))$ where $x \in \mathcal{X}$, $s \in \mathrm{T}(\mathcal{G}, \mathcal{X})$ and e is a set of equations between terms of $\mathrm{T}(\mathcal{G}, \mathcal{X})$. The meaning of a step is that upon receiving message x, the honest agent checks the equations in e and sends the message s. An execution of a protocol is a finite sequence of steps.

Example 1. *Consider the following toy protocol, where K is a symmetric key initially known by A only:*

$$A \rightarrow B : \{M \oplus B\}_K$$
$$B \rightarrow A : B$$
$$A \rightarrow B : K$$
$$B \rightarrow A : M$$

Assuming the algebraic properties of $\oplus$, symmetric encryption $\mathrm{se}(\_, \_)$ *and symmetric decryption* $\mathrm{sd}(\_, \_)$*, we model this protocol as:*

$\text{RECV}(v_1); \text{SEND}(\text{se}(M \oplus B, K)); \text{COND}(v_1 = \text{c}_{\min})$
$\text{RECV}(v_2); \text{SEND}(B); \text{COND}(\emptyset)$
$\text{RECV}(v_3); \text{SEND}(K); \text{COND}(v_3 = B)$
$\text{RECV}(v_4); \text{SEND}(\text{sd}(v_2, v_4) \oplus B); \text{COND}(v_2 = \text{se}(x, v_4,))$
$\text{RECV}(v_5); \text{SEND}(\text{c}_{\min}); \text{COND}(v_5 = M)$

Note that in our setting we can model that at some step i the message must match the pattern t_i by adding an equation $v_i \stackrel{?}{=} t_i$ as a condition for this step.

In order to define whether an execution of a protocol is feasible, we must first define when a substitution σ satisfies a set of equations $\mathcal{S}$.

Definition 3. *(Unification systems) Let $\mathcal{H}$ be a set of axioms on $\mathrm{T}(\mathcal{G}, \mathcal{X})$. An $\mathcal{H}$-*Unification system *$\mathcal{S}$ is a finite set of equations in $\mathrm{T}(\mathcal{G}, \mathcal{X})$ denoted by $(t_i \stackrel{?}{=} u_i)_{i \in \{1,\ldots,n\}}$. It is satisfied by a ground substitution σ, and we write $\sigma \models \mathcal{S}$, if for all $i \in \{1, \ldots, n\}$ $t_i\sigma =_{\mathcal{H}} u_i\sigma$.*

Let $\mathcal{I}{=}\langle \mathcal{G}, T, \mathcal{H} \rangle$ be an intruder system. A *configuration* is a couple $\langle P, N \rangle$ where P is a finite sequence of steps and N is a set of ground terms (the knowledge of the intruder). From the configuration $\langle (\text{RECV}(x); \text{SEND}(s); \text{COND}(e)) \cdot P, N \rangle$ a transition to (P', N') is possible iff there exists a ground substitution σ such that $x\sigma \in \overline{N}$, $\sigma \models e$, $N' = N \cup \{s\sigma\}$ and $P' = P\sigma$. Trace based-security properties like secrecy can be reduced to the *Execution feasibility* problem:

Execution feasibility

Input:	an initial configuration $\langle P, N_0 \rangle$
Output:	SAT iff there exists a reachable configuration $\langle \emptyset, M \rangle$

3.4.3 Constraints systems

We express the execution feasibility of a protocol by a constraint system $\mathcal{C}$.

Definition 4. *(Constraint systems) Let $\mathcal{I} = \langle \mathcal{G}, T, \mathcal{H} \rangle$ be an intruder system. An $\mathcal{I}$-*constraint system *$\mathcal{C}$ is denoted by $((E_i \rhd v_i)_{i \in \{1,\ldots,n\}}, \mathcal{S})$, and it is defined by a sequence of couples $(E_i, v_i)_{i \in \{1,\ldots,n\}}$ with $v_i \in \mathcal{X}$ and $E_i \subseteq \mathrm{T}(\mathcal{G}, \mathcal{X})$ for $i \in \{1, \ldots, n\}$ and $E_{i-1} \subseteq E_i$ for $i \in \{2, \ldots, n\}$, and by an $\mathcal{H}$-unification system $\mathcal{S}$. It is* deterministic *iff $\mathrm{Var}(E_i) \subseteq \{v_1, \ldots, v_{i-1}\}$ for all $i \in \{1, \ldots, n\}$.*

*An $\mathcal{I}$-*constraint system *$\mathcal{C}$ is satisfied by a ground substitution σ if for all $i \in \{1, \ldots, n\}$ we have $v_i\sigma \in \overline{E_i\sigma}$ and if $\sigma \models \mathcal{S}$. We denote that a ground substitution σ satisfies a constraint system $\mathcal{C}$ by $\sigma \models_{\mathcal{I}} \mathcal{C}$.*

Constraint systems are denoted by $\mathcal{C}$ and decorations thereof. Note that if a substitution σ is a solution of a constraint system $\mathcal{C}$, by definition of constraints and of unification systems the substitution $(\sigma)\downarrow_{\mathcal{O}}$ is also a solution of $\mathcal{C}$ (where $\mathcal{O}$ is an o-completion of H). In the context of security protocols, the inclusion $E_{i-1} \subseteq E_i$ means that the knowledge

of an intruder does not decrease as the protocol progresses: after receiving a message an honest agent will respond to it. This response can be added to the knowledge of an intruder who listens to all communications.

The condition defining the *deterministic* constraint systems expresses that a message to be sent at some step i should be built from previously received messages recorded in the variables v_j for $j < i$ and from the initial knowledge.

Example 2. *We model the protocol of Example 1 by the following constraint system. First we gather all conditions in a unification system* $\mathcal{S}$

$$\mathcal{S} = \left\{ v_1 \stackrel{?}{=} \mathrm{c_{min}} \ , \ v_3 \stackrel{?}{=} B \ , \ v_2 \stackrel{?}{=} \mathrm{se}(x, v_4) \ , \ v_5 \stackrel{?}{=} M \right\}$$

The protocol execution for intruder $\mathcal{I}$ *with initial knowledge* $\{\mathrm{c_{min}}\}$ *is then expressed by the constraint system:*

$$\begin{aligned} \mathcal{C} = ((\ & \mathrm{c_{min}} \rhd v_1, \\ & \mathrm{c_{min}}, \mathrm{se}(M \oplus B, K) \rhd v_2, \\ & \mathrm{c_{min}}, \mathrm{se}(M \oplus B, K), B \rhd v_3, \\ & \mathrm{c_{min}}, \mathrm{se}(M \oplus B, K), B, K \rhd v_4), \\ & \mathrm{c_{min}}, \mathrm{se}(M \oplus B, K), B, K, \mathrm{sd}(v_2, v_4) \oplus B \rhd v_5, \mathcal{S}) \end{aligned}$$

The deterministic *condition imposes to write the last message* $\mathrm{sd}(v_2, v_4)$ *instead of* x *though both are equivalent with respect to satisfiability.*

The decision problems we are interested in are the *satisfiability* and the *ordered satisfiability* of intruder constraint systems.

Satisfiability

Input: an $\mathcal{I}$-constraint system $\mathcal{C}$
Output: SAT iff there exists a substitution σ such that $\sigma \models_{\mathcal{I}} \mathcal{C}$.

In order to be able to combine solutions of constraints in component theories to get a solution for the full theory, these solutions have to satisfy some ordering constraints too. Intuitively, this is to avoid introducing cycles when building a global solution. With respect to this use we can always assume $\mathrm{c_{min}}$ is the minimum of $\prec$ in the following definition:

Ordered Satisfiability

Input: an $\mathcal{I}$-constraint system $\mathcal{C}$, X the set of all variables and C the set of all free constants occurring in $\mathcal{C}$ and a linear ordering $\prec$ on $X \cup C$.
Output: SAT iff there exists a substitution σ such that $\sigma \models_{\mathcal{I}} \mathcal{C}$ and $x \prec c$ implies $c \notin \mathrm{Sub_{syn}}(x\sigma)$ for all $x \in X$ and for all $c \in C$.

Our main result is the following modularity theorem, which is obtained by designing an algorithm for solving $\mathcal{U}$-constraints using algorithms for solving *ordered satisfiability* for intruders $\langle \mathcal{F}_1, T_1, \mathcal{E}_1 \rangle$ and $\langle \mathcal{F}_2, T_2, \mathcal{E}_2 \rangle$.

Theorem 1. *If the ordered satisfiability problem is decidable for two intruders* $\langle \mathcal{F}_1, T_1, \mathcal{E}_1 \rangle$ *and* $\langle \mathcal{F}_2, T_2, \mathcal{E}_2 \rangle$ *for disjoint signatures* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *then the satisfiability problem is decidable for* deterministic *constraint systems for the intruder* $\mathcal{U} = \langle \mathcal{F}, \langle T_1 \rangle \cup \langle T_2 \rangle, \mathcal{E} \rangle$.

In order to implement these results on combining intruder theories in the AVISPA Tool we have to tackle two problems. First, the constraint solvers for subtheories of interest have to be designed. Our experience in developing unification algorithms should be quite helpful with respect to this task. In fact we have been able to design constraint solving procedure for most theories of interest for protocols: xor, exponentiation, abelian groups, pairing, abstract encryption and decryption. The only limitation we have is that the protocols have to be expressed using explicit destructors like in the applied pi calculus. Second, we have to translate the protocol description in HLPSL to descriptions that are suitable for analysis and are similar to applied pi calculus ones. This should be easily done using a translation technique developed by Gotsman, Massacci, and Pistore [39], who have recently published a procedure to perform the translation.

4 Abstract Secure Communication Channels

4.1 Motivation and Contribution

Like most protocol specification languages (such as muCAPSL [46]), the High Level Protocol Specification Language HLPSL of the AVISPA Tool [6, 20] restricts its attention to a single intruder model: the Dolev-Yao intruder model [35]. This is the most powerful intruder model, and is generally the type of intruder which protocols are designed to be secure against. The restriction to a single intruder model has provided modellers with a simple way to model communication without having to specify all the actions the intruder might take. While this has proved to be useful as it allowed us (and others) to analyse a very large number of industrial-scale Internet protocols, it is however also limiting as more and more protocols are being proposed for operating in different environments and over new types of media. It is thus necessary to find means of analysing these protocols, and even just parts of these protocols, with respect to different models of an intruder's capabilities. One example is wireless communication, where the intruder may not necessarily be able to block selected messages.

Another important observation about protocol compositionality is that the analysis of industrial protocols that make use of sub-protocols can be a tedious task. It involves manually combining several protocol specifications into one by modelling the message exchanges of the sub-protocols in the appropriate places in the main protocol. When a protocol makes use of a number of sub-protocols, the specifications can become quite complex and there is a high chance of mistakes occurring in the specification. Moreover, sometimes a

protocol will make use of sub-protocols, but not actually specify the protocols to use. In this case, modellers (and vendors) must choose an appropriate sub-protocol to model (or build) based on the assumptions the main protocol makes about the sub-protocol. The modelling process can be simplified by a mechanism with which to model the assumptions a protocol makes about a sub-protocol, instead of actually modelling the sub-protocol.

Furthermore, HLPSL cannot be used to model non-repudiation protocols or security protocols with non-repudiation requirements. The language provides no way to specify which actions generate proof evidence, and thus there is no way to reason about an agent's ability to prove or deny its actions or the actions of other agents. Non-repudiation properties are an important part of e-commerce protocols and exchange protocols and this lack of support reduces the applicability of the AVISPA Tool.

We report here on work that has been carried out by Daniel Plasto [49] and the SIEMENS project partner on introducing *(Abstract) Secure Communication Channels (SCC)* into HLPSL proposing syntax, semantics and tool implementation for different types of SCC. This research also investigates the effectiveness of SCC via specification experiments, modelling complex protocols using a top-down approach.

4.1.1 Abstract Secure Communication Channels (SCC)

Secure Communication Channels are used to model properties of a communication medium or a protocol layer that are beyond the scope of the analysis. The channel allows the modeller to specify properties of the communication over this medium. These properties are at a lower level of abstraction than the protocol specification and are thus abstractly expressed using a channel type. This is not restricted to properties of the physical layer of a protocol but, for instance, a channel might provide the privacy security property to model a secure channel based on TLS or IPsec. Thus, SCC can be used when modelling larger application protocols which make use of several sub-protocols. The sub-protocols can be modelled as SCCs which provide certain security properties.

4.1.2 Advantages of SCC

The key advantage of a channel-based approach for specifying assumptions about communication is that channels provide a high level of flexibility. By creating a number of different named channels, each of a different type, it is possible to model situations where some messages are sent using one communication model and others are sent using different models of communication. This capability is extremely important for some of the protocols under development. For example, [25] and [10] both discuss a scenario where a user wishes to confidentially send a document to a public printer from a PDA. A physical connection is used to establish a keyed link between the PDA and the printer, negating the need for a Public Key Infrastructure. This situation is impossible to model without the ability to specify different intruder models for different parts of a protocol (i.e. channels).

Another advantage of the channel approach is that it enables a top-down approach to protocol design and analysis, where protocol designers first specify their needs, and

can then incrementally instantiate a protocol with a concrete realisation of those needs. Alternatively, after specifying the needs of a sub-protocol, the designers may search for a protocol they can use which provides the properties they require, and perhaps even analyse this protocol separately.

4.2 SCC in the HLPSL Language

As part of this research, we have extended the HLPSL language to include the new channel types and message parameters. These extensions allow modellers to make use of SCC: they are declared as constants in the same way as other shared information is declared. Each channel is given a name and is of type channel, but must additionally be given a channel type attribute which defines its semantics.

We have defined the semantics of each of the new channel types in the IF term-rewriting language [7]. Channels behave in a way similar to that of the intruder. Messages sent to a channel are remembered forever, and may be sent on at an arbitrary time, or not at all. To implement the functionality into the AVISPA Tool, we have correspondingly modified and extended the HLPSL2IF Translator (which translates specifications written in HLPSL into specifications written in IF) to capture the required behaviour of each channel and thus support these new constructs, and we have extended the IF prelude file with new facts used to reason about SCC. These facts are used to describe the knowledge of channels, as well to describe the ability of agents to prove who has sent and received certain messages. Although it is possible to declare these facts in the signature section of each IF file, it makes more sense to add them to the IF prelude file, which is used to describe protocol independent functions and facts.

Note that the scope of the changes made to the tools was limited to the translation process. The HLPSL language and the HLPSL2IF Translator have also been extended to support message parameters, which are used when sending and receiving messages on SCC. Finally, the translator has been modified so that send and receive actions on these channels are translated into appropriate IF rules. The IF specifications generated by the new version of the translator are valid IF files and no changes need to be made to the back-ends of the AVISPA Tool. Making modifications to the input language of all four back-ends would break the compatibility between the HLPSL2IF Translator and the back-ends.

4.3 Types of SCC

SCCs are channels that behave differently from Dolev-Yao channels. In some cases, messages are not sent to the intruder and the channel itself is used to store messages, while in other cases reception will be triggered by the channel instead of the intruder. Moreover, some channels generate special proof facts which can be used when specifying the channel requirements. The following is a summary of the new types of communication channels:

Named Channels In SCCs, we assume that agents must have explicitly access to the same channel in order to communicate over it.

Over-the-air Channels Over-the-air channels deliver the message to the intruder at the same time as they deliver the message to the recipient. This is used to model a wireless communication medium in which it is difficult for the intruder to intercept messages on-the-fly. When an agent sends a message on an over-the-air channel, it is received by the intruder at the same time as it is received by the intended recipient. It is possible for the intruder to send messages over an over-the-air channel; however, the intruder's ability to manipulate a protocol on-the-fly, such as in man-in-the-middle attacks is limited by this communication model. Notice that there are two possible receive actions, one in which the message is received from the channel by the recipient and the intruder, and another where the intruder tricks the recipient into accepting another message. Some models of over-the-air communication stipulate that the intruder cannot block messages, and thus all messages are eventually delivered to the recipient. As the current version of the IF cannot support fairness constraints such as *eventually*, this functionality is not yet included.

Authentic Channels As well as being a common type of security protocol, authentication protocols are commonly used as sub-protocols for more complex protocols. Therefore, we have included in SCCs also an authentic communication channel for sending and receiving messages in an authentic way. When a message is sent over an authentic channel, the knowledge of the channel is updated with the message and the knowledge of the intruder is updated. When an agent receives a message on an authentic channel, it specifies the expected sender of the message. For an agent to receive a message on an authentic channel it is necessary that indeed the exact message was previously sent by the expected sender. The channel presented here provides weak authentication. Strong authentication additionally requires replay protection, i.e. that a message sent by an agent Alice is never received by another agent Bob more than once.

Confidential Channels Confidential channels are also commonly established in sub-protocols, so that the main protocol can then assume that the involved parties have established a secure channel with which they can communicate privately. When sending on a confidential channel, the knowledge of the intruder is not updated whilst the knowledge of the channel is. A receive event on a confidential channel is enabled if an appropriate message has been sent on the channel, or if the intruder has knowledge of an appropriate message. The intruder may be able to construct a message that the receiver will accept, even though he does not receive the messages being sent.

Non-repudiation of Origin Channels A number of protocols, particularly e-commerce protocols, have requirements involving non-repudiation of origin. In real life, non-repudiation may be implemented by a variety of means, including signatures, trusted computing bases, trusted third parties, or detailed secure logs. SCCs can be used to indicate which messages generate which types of information that may be used as evidence (or "proof") that a certain action happened. Security requirements can then express non-repudiation properties based on the proofs generated by these channels.

The non-repudiation of origin property states that an agent may not deny sending a particular message. In other words, the recipient can prove that the sender sent the message. A non-repudiation of origin channel is identical to an authentication channel except that upon receipt of a message over a non-repudiation of origin channel, the recipient is given the ability to prove that the sender really sent the message. All channels that provide a non-repudiation of origin are also authentic channels: if an agent is to be provided with proof that a certain message was sent or received by another agent, then the message must have actually been sent or received by that agent.

Non-repudiation of Receipt Channels Non-repudiation of receipt means that the sender of a message obtains some proof that the recipient did receive a particular message. Non-repudiation of receipt is not easily implemented in concrete protocols (and there is no clear consensus on how to do it). However, SCCs allow modellers to specify this property without worrying about the details of the sub-protocol implementing it. A non-repudiation of receipt channel is an authentic channel that generates a proof when messages are received on the channel. This proof allows the sender to prove that the recipient did receive the message.

4.4 Experimentation

In order to determine how useful the introduced concepts are to protocol modellers, we chose three examples of real protocol which are difficult to model with the tools available today: Purpose Built Keys Framework (PBK [15]), the Asokan-Shoup-Waidner Contract Signing Protocol (ASW [4, 5]), and the Internet Open Trading Protocol (IOTP [17]).

These case studies allowed us to evaluate the worth of the new channel types. These examples demonstrate how SCC can simplify the work of protocol modellers by exploiting the increased expressive capabilities of the proposed protocol specification language. For instance, in BPK a channel is simply assumed to be authentic, instead of having to secure explicitly the message exchanges over the channel. The ASW protocol is modelled by simply assuming that special channels generate information (modelled as facts) that is to be interpreted as proof that a certain message was indeed sent by the claimed sender. This fact is available to the receiver of the message and may be used in the rest of the protocol.

The PBK framework makes an assumption: that the first message of the protocol was received intact, by some reason that is beyond the scope of the discussion. This may be because the principals were lucky or because they are using a "safe Route", where most probably there will be no attacker, or because the first message is secured in some way, possibly by replication or broadcast, but not necessarily through a specific cryptographic mechanism. *If* the first message is received intact *then* we claim a set of security properties. Prior to SCC this was modelled by giving each party a shared key and encrypting the message, which is not a faithful modelling. Using SCCs, the assumption was modelled by sending the first message over an authentic channel instead of assuming the existence of a shared key.

The ASW protocol presented a challenge because it has non-repudiation assumptions and related requirements. This protocol had previously been modelled, but the non-repudiation properties of the protocol were not checked.

The modelling of ASW is at a higher abstraction layer than the original one in [4, 5]. Instead of explicitly modelling signatures and arguing that those signatures can not be forged by the intruder (which is not the goal of the protocol, but simply an assumption: the corresponding private keys are never sent around), we model the corresponding actions as happening via a non-repudiation channel (denoted by channel (nro)). Practically that means that any other means of non-repudiation mechanism (e.g. secure trusted platforms or secure logging via a notary service) could be used instead. The non-repudiation property of the ASW protocol is defined in [4, 5] as: after an effective exchange, (i.e. `P` has received `iQ` at the end of the exchange), `P` will be able to prove that `iQ` originated from `Q`, and that `Q` received `iP`. This can be directly written in IF:

```
goal nro_1(O,Me1,No,Nr,R,Me2) := accepted(O).accepted(R).
received(O,pair(Me1,pair(No,pair(Me2,Nr))),pair(Me1,pair(No,pair(Me2,Nr)))).
received(R,pair(Me1,pair(No,pair(Me2,Nr))),pair(Me1,pair(No,pair(Me2,Nr)))).
not(canprovesent(O,R,Me2)
```

The `canprovesent` fact has the following form:

```
canprovesent: agent * agent * message -> fact
```

It is generated whenever an agent gains the ability to prove that another agent sent a certain message, which is a fact generated when an agent receives a message via a non-repudiation channel. In other words, sending messages over non-repudiation channels are actions that generate an information that can be used (by the recipient) as evidence for non-repudiation. We call this piece of information a "proof".

The non-repudiation properties of the ASW protocol were then specified in terms of the proof facts generated by the channels and the non-repudiation-related goals of the protocol were validated. More precisely, we showed that in one single run of the protocol (that has many ramifications and choices for the both partners) we can safely conclude that it is impossible to arrive to a position in which he has advantage over the other, by receiving his expected share but the other one not. This result was not possible using HLPSL prior to the addition of SCC.

The IOTP protocol provides a framework into which sub-protocols are incorporated. The purchase transaction of the IOTP protocol was specified using SCC to model the authentication sub-protocol. This allows modellers to concentrate on the IOTP protocol itself, rather than spending time choosing a particular sub-protocol and modelling it as part of the main protocol. The IOTP purchase transaction was successfully analysed (for one session, similar to the ASW protocol) using an authentic channel to model the authentication sub-protocol. See [49] for further details.

5 Conclusion

We have described three different approaches that we have investigated to tackle different aspects of compositionality: a general method for reasoning about contract-signing protocols using a specialised protocol logic, an algorithm for combining decision procedures for intruder constraints on disjoint signatures, and (Abstract) Secure Communication Channels as a mean of modelling larger application protocols which make use of several subprotocols and where one wishes to specify different intruder models for different parts of a protocol.

Our experimental analyses show that these methods are very useful to further scale up the automated deduction techniques and tools that we have been developing in the AVISPA project.

As future work, we have begun developing an integrated environment consisting of automated reasoning back-end tools for ensuring global security of composed security services based on security protocols and their complex orchestration. As a concrete example, we have started applying our tools to the analysis of complex Web Services and the first experiences are very positive.

References

[1] M. Abadi, B. Blanchet, and C. Fournet. Just Fast Keying in the Pi Calculus. In *Proceedings of the 13th European Symposium on Programming (ESOP'04)*, LNCS 2986, pages 340–354. Springer, 2004.

[2] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proc. 10th International Computer Aided Verification Conference*, pages 521–525, 1998.

[3] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In C. Palamidessi, editor, *Proceedings of Concur'00*, LNCS 1877, pages 380–394. Springer-Verlag, 2002.

[4] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. Technical Report RZ 2976, IBM Research, 1997.

[5] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99. IEEE, 1998.

[6] AVISPA. Deliverable 2.1: The High-Level Protocol Specification Language. Available at `http://www.avispa-project.org/publications.html`, 2003.

[7] AVISPA. Deliverable 2.3: The Intermediate Format. Available at `http://www.avispa-project.org/publications.html`, 2003.

[8] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories. combining decision procedures. *J. Symb. Comput.*, 21(2):211–243, 1996.

[9] M. Backes, A. Datta, A. Derek, J. Mitchell, and M. Turuani. Compositional analysis of contract signing protocols. In *Proceedings of 18th IEEE Computer Security Foundations Workshop*, 2005.

[10] D. Balfanz, D. K. Smetters, P. Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of Network and Distributed System Security Symposium 2002 (NDSS'02)*, 2002.

[11] D. Basin, S. Mödersheim, and L. Viganò. Ofmc: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, June 2005. Published online December 2004.

[12] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of the 28th International Conference on Automata, Language and Programming: ICALP'01*, LNCS 2076, pages 667–681. Springer-Verlag, Berlin, 2001.

[13] M. Boreale and M. Buscemi. Symbolic analysis of crypto-protocols based on modular exponentiation. In *Proceedings of MFCS '03*, LNCS 2747. Springer-Verlag, Berlin, 2001.

[14] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking (MOBICOM-01)*, pages 180–188. ACM Press, 2001.

[15] S. Bradner, A. Mankin, and J. I. Schiller. A framework for purpose built keys (PBK). Internet draft, November 2002.

[16] M. Bugliesi, R. Focardi, and M. Maffei. Compositional analysis of authentication protocols. In *Proceedings of ESOP'04*, LNCS 2986, pages 140–154. Springer-Verlag, 2004.

[17] D. Burdette and M. Goncalves. Internet Open Trading Protocol, 2000.

[18] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/`.

[19] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In *8-th ACM Conference on Computer and Communications Security*, pages 176–185. ACM Press, 2001.

[20] Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Automated Software Engineering. Proceedings of the Workshop on Specification and Automated Processing of Security Requirements, SAPS'04*, pages 193–205. Austrian Computer Society, Austria, September 2004.

[21] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In P. Kolaitis, editor, *Proceedings of LICS'2003*. IEEE, 2003.

[22] Y. Chevalier and M. Rusinowitch. Combining Intruder Theories. Technical report, INRIA — extended abstract to appear in the proceedings of ICALP'05, 2005. `http://www.inria.fr/rrrt/rr-5495.html`.

[23] Y. Chevalier and L. Vigneron. A Tool for Lazy Verification of Security Protocols. In *Proceedings of ASE'01*. IEEE Computer Society Press, 2001.

[24] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. Symp. on Logic in Computer Science (LICS'03), IEEE Computer Society Press*, 2003.

[25] S. Creese, M. Goldsmith, and B. Roscoe. Authentication for pervasive computing. In *Proceedings of the International Conference on Security in Pervasive Computing*, pages 116–129, 2003.

[26] A. Datta, A. Derek, J. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proceedings of the 19th Annual Conference on Mathematical Foundations of Programming Semantics*, Electronic Notes in Theoretical Computer Science 83. Elsevier Science, 2004.

[27] A. Datta, A. Derek, J. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security (Special Issue of Selected Papers from CSFW-16)*, to appear.

[28] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *Proceedings of 16th IEEE Computer Security Foundations Workshop*, pages 109–125. IEEE, 2003.

[29] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition (Extended abstract). In *Proceedings of ACM Workshop on Formal Methods in Security Engineering*, pages 11–23, 2003.

[30] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proceedings of 17th IEEE Computer Security Foundations Workshop*, pages 30–45. IEEE, 2004.

[31] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proceedings of 19th Annual Conference on Mathematical Foundations of Programming Semantics*, volume 83 of *Electronic Notes in Theoretical Computer Science*, 2004.

[32] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, to appear.

[33] S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, Washington, D.C., USA, Oct. 2004. ACM Press.

[34] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B*, pages 243–320. Elsevier, 1990.

[35] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

[36] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *Proceedings of 14th IEEE Computer Security Foundations Workshop*, pages 241–255. IEEE, 2001.

[37] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11:677–721, 2003.

[38] J. A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Advances in Cryptology: Proceedings of Crypto'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer-Verlag, 1999.

[39] A. Gotsman, F. Massacci, and M. Pistore. Towards an Independent Semantics and Verification Technology for the HLPSL Specification Language. *Electronic Notes in Theoretical Computer Science 135(1):59–77 (Proceedings of the Workshop on Automated Reasoning for Security Protocol Analysis, ARSPA 2005)*, 2005.

[40] J. D. Guttman. Authentication tests and disjoint encryption: A design method for security protocols. *Journal of Computer Security*, 12(3/4):409–434, 2004.

[41] J. D. Guttman and F. J. Thayer Fábrega. Protocol independence through disjoint encryption. In *Proc. CSFW'00*. IEEE Computer Society Press, 2000.

[42] T. A. Henzinger, S. Qadeer, and S. K. Rajamani. You assume, we guarantee: Methodology and case studies. In *Proc. 10th International Computer Aided Verification Conference*, pages 440–451, 1998.

[43] J. Hsiang and M. Rusinowitch. On word problems in equational theories. In *ICALP*, volume 267 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 1987.

[44] K. L. McMillan. A compositional rule for hardware design refinement. In *Proc. 9th International Computer Aided Verification Conference*, pages 24–35, 1997.

[45] C. Meadows and P. Narendran. A unification algorithm for the group diffie-hellman protocol. In *Workshop on Issues in the Theory of Security (in conjunction with POPL'02), Portland, Oregon, USA, January 14-15,*, 2002.

[46] J. Millen and G. Denker. MuCAPSL. In *Proceedings of DISCEX III, DARPA Information Survivability Conference and Exposition*, pages 238–249. IEEE Computer Society Press, 2003.

[47] J. Millen and V. Shmatikov. Symbolic protocol analysis with an abelian group operator or Diffie-Hellman exponentiation. *Journal of Computer Security*, 2005.

[48] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the ACM Conference on Computer and Communications Security CCS'01*, pages 166–175, 2001.

[49] D. Plasto. *Automated Analysis of Industrial Scale Security Protocols.* Phd, School of IT, Bond University, Gold Coast, Queensland 4229, Australia, November 2004.

[50] M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *Proceedings of CSFW'01.* IEEE Computer Society Press, 2001. Available at `http://www.avispa-project.org`.

[51] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *J. Symb. Comput.*, 8(1/2):51–99, 1989.

[52] V. Shmatikov and J. C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science*, 283(2):419–450, 2002.