Synthesizing secure protocols

Véronique Cortier, Bogdan Warinschi, Eugen Zalinescu

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Synthesizing secure protocols

Véronique Cortier — Bogdan Warinschi — Eugen Zălinescu

N° ????

Avril 2007

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

Synthesizing secure protocols

Véronique Cortier*, Bogdan Warinschi† , Eugen Zălinescu*

Thème SYM — Systèmes symboliques
Projets Cassis

Rapport de recherche n° ???? — Avril 2007 — 32 pages

Abstract: We propose a general transformation that maps a protocol secure in an extremely weak sense (essentially in a model where no adversary is present) into a protocol that is secure against a fully active adversary which interacts with an *unbounded* number of protocol sessions, and has absolute control over the network. The transformation works for arbitrary protocols with any number of participants, written with usual cryptographic primitives. Our transformation provably preserves a large class of security properties that contains secrecy and authenticity.

An important byproduct contribution of this paper is a modular protocol development paradigm where designers focus their effort on an extremely simple execution setting – security in more complex settings being ensured by our generic transformation. Conceptually, the transformation is very simple, and has a clean, well motivated design. Each message is tied to the session for which it is intended via digital signatures and on-the-fly generated session identifiers, and prevents replay attacks by encrypting the messages under the recipient's public key.

Key-words: security protocols, signatures, public-key encryption

* LORIA, CNRS & INRIA project Cassis, Nancy, France. This work has been partly supported by the ACI Jeunes Chercheurs JC 9005 and the ACI Satin
† Computer Science Department, University of Bristol

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 —Télécopie : +33 3 83 27 83 19

Synthèse de protocoles sûrs

Résumé : Nous proposons une transformation générale pour synthétiser des protocoles sûrs par construction. Notre transformation part d'un protocole sûr en un sens très faible (le protocole doit être sûr simplement lorsqu'il est exécuté une seule fois et sans adversaire) et produit un protocole sûr contre un adversaire contrôlant tous les échanges de messages sur le réseau durant un nombre *illimité* de sessions. Cette transformation fonctionne pour des protocoles arbitraires, avec un nombre quelconque de participants et des primitives cryptographiques usuelles. Nous prouvons la sécurité des protocoles synthétisés pour une large classe de propriété incluant le secret et l'authentification.

Mots-clés : protocoles de sécurité, signatures, chiffrement publique

Contents

1 Introduction

Cryptographic protocols are small programs designed to ensure secure communications over an untrusted network. Their security is of crucial importance due to their widespread use in critical systems and in day-to-day life. Unfortunately, designing and analyzing such protocols is a notoriously difficult and error-prone task, largely due to the potentially unbounded behavior of malicious agents.

In this paper we contribute to a popular technique that has been developed to cope with this problem. Under the paradigm that we study, one can start with the design of a simple version of a system intended to work in restricted environments (*i.e.* with restricted adversaries) and then obtain, via a generic transformation, a much stronger system intended to work in arbitrary environments. More specifically, we introduce one such transformation that takes as input a protocol that is secure (in a sense that we discuss below) in a *single* execution of the protocol, with *no adversary* present (not even a passive eavesdropper). The output of the transformation is a protocol that withstands a realistic adversary with absolute control of the communication between an unbounded number of protocol sessions. The details of our transformation are useful to understand how security is translated from the simple to the more complex setting.

Our transformation. At a high level, the transformation works by bounding messages to sessions using digital signatures on messages concatenated with dynamically generated session identifiers, and hiding messages from the adversary using public key encryption. More specifically, the transformation is as follows. Consider a protocol with k participants $A_1, \ldots, A_k$ and n exchanges of messages.

$$\begin{aligned} A_{i_1} &\rightarrow A_{j_1} : \quad m_1 \\ &\vdots \\ A_{i_n} &\rightarrow A_{j_n} : \quad m_n \end{aligned}$$

The transformed protocol starts with a preliminary phase, where each participant broadcasts a fresh nonce to all others participants. Intuitively, the concatenation of the nonces with the identities of the participants forms a session identifier.

$$\begin{aligned} A_1 &\rightarrow \mathsf{All} : \quad N_1 \\ &\vdots \\ A_k &\rightarrow \mathsf{All} : \quad N_k \end{aligned}$$

Let $\mathsf{sessionID} = \langle A_1, A_2, \ldots, A_k, N_1, N_2, \ldots, N_k \rangle$. The remainder of the protocol works roughly as the original one except that each message is sent together with a signature on the message concatenated with the session identifier, and the whole construct is encrypted

under the recipient's public key:

$$\begin{aligned} A_{i_1} \to A_{j_1} : \quad & \{\![m_1, [\![m_1, p_1, \mathsf{sessionID}]\!]_{\mathsf{sk}(A_{i_1})}]\!\}_{\mathsf{pk}(A_{j_1})} \\ \vdots & \\ A_{i_n} \to A_{j_n} : \quad & \{\![m_n, [\![m_n, p_n, \mathsf{sessionID}]\!]_{\mathsf{sk}(A_{i_n})}]\!\}_{\mathsf{pk}(A_{j_n})} \end{aligned}$$

where the p_i's are the current control points in the participant's programs. We write $[\![m]\!]_{\mathsf{sk}(A)}$ for the message m tied with its signature with the signing key of A and $\{\![m]\!\}_{\mathsf{pk}(A)}$ for the encryption of m under the public encryption key of A.

Security preservation. Intuitively, our transformation ensures that the messages of the protocol sent between honest parties in any given session of the original protocol, cannot be learned and/or blindly replayed by the adversary to unsuspecting users in other protocol sessions. Indeed, the adversary cannot impersonate users in honest sessions (since in this case it would need to produce digital signatures on their behalf), and cannot learn secrets by replaying messages from one session to another (since messages are encrypted, and any blindly replayed message would be rejected due to un-matching session identifiers).

It is essentially this property that ensures that security properties that hold in single executions with no adversary present also hold in the stronger, fully active adversary model. Although the transformation does not preserve all imaginable security properties (for example, any anonymity that the original protocol might enjoy is lost due to the use of public key encryption) it does preserve several interesting properties. In particular, we exhibit a class of logic formulas which, if satisfied in single executions of the original protocol are also satisfied by the transformed protocol in the presence of active adversaries. The class that we consider includes standard formulations for secrecy and authentication (for example injective agreement [13] and several other variants).

Simple protocol design. Our transformation enables more modular and manageable protocol development. One can start by building a protocol with the desirable properties built-in, and bearing in mind that no adversary is actually present. Then, the final protocol is obtained using the transformation that we propose. We remark that designers can easily deal with the case of single session and it is usually the more involved setting (multi-party, many-session) that causes the real problems. As an example, we show how to derive a simple protocol for authentication later in the paper. Our transformation can be applied to any kind of protocols, with any number of participants (although, the number of participants in each session should not be too large for efficiency).

Simple protocol verification. In standard protocol design, the potentially unbounded behavior of malicious agents makes verification of protocols is an extremely difficult task. Even apparently simple security properties like secrecy are undecidable in general [8]. One obvious approach to enable verifiability is to consider restrictions to smaller protocol classes. For example, it can be shown that for finite number of parallel sessions secrecy preservation

is co-NP-complete [16]. Most automatic tools are based on this assumption, which is often sufficient to discover new attacks but does not allow in general to prove security properties. It is also possible to ensure verifiability of protocols even for unbounded number of sessions by restricting the form of messages, and/or the ability to generate new nonces, e.g. [8, 3, 4], but only a few such results do not make this unreasonably strong assumption [14, 15].

For the class of protocols obtained via our transformation, security verification is significantly trivialised. Indeed, for a *single, honest* execution, security is in fact closer to correctness, and should be easily carried out automatically.

Related work. The kind of modular design paradigm that we propose is rather pervasive in cryptographic design. For example, Goldreich, Micali, and Wigderson show how to compile arbitrary protocols secure against participants that honestly follow the protocol (but may try to learn information they are not entitled to) into protocols secure against participants that may arbitrarily deviate from the protocol [9]. Bellare, Canetti, and Krawczyk have shown how to transform a protocol that is secure when the communication between parties is authenticated into one that remains secure when this assumption is not met [2]. All of the above transformations have a different goal, apply to protocols that need to satisfy stronger requirements than ours, and are also different in their design.

Our work is inspired by a recent compiler introduced by Katz and Yung [11] which transforms any group key exchange protocol secure against a passive adversary into one secure against an active adversary. Their transformation is, in some sense, simpler since they do not require that the messages in the transformed protocol are encrypted. However, their transformation is also weaker since although it requires that the protocol be secure against passive adversaries, these adversaries still can corrupt parties adaptively (even after the execution has finished). Furthermore, while their transformation is sufficient for the case of group key exchange, it fails to guarantee the transfer of more general security properties. The reason for the failure is that an adversary can obtain a message (e.g. a ciphertext) from a session with only honest participants, and get information about the message (e.g. the underlying plaintext) by replaying it in some other sessions for which he can produce the necessary digital signatures. We further discuss and compare the two transformations via an illustrative example in Section 2.

Our transformation might be viewed as a way of transforming protocols into fail-stop protocols, introduced by Gong and Syverson [10], where any interference of an attacker is immediately observed and causes the execution to stop. But for fail-stop protocols, it is still necessary to consider the security issues related to the presence of passive adversaries. Here we achieve more since we obtain directly secure protocols. Moreover, a major difference is that we provide formal proof of the security of the resulting protocols while the approach of [10] is rather a methodology for prudent engineering. In particular, there are no proved guarantees on the security of the resulting protocols.

Datta, Derek, Mitchell, and Pavlovic [7] propose a methodology for modular development of protocols where security properties are *added* to a protocol through generic transformations. In contrast, our transformation starts from protocols where the security property is

built-in. Abadi, Gonthier, and Fournet give a compiler for programs written in a language with abstractions for secure channels into an implementation that uses cryptography [1] and is similar to ours in the sense that it aims to eliminate cryptographic security analysis in involved settings. However the overall goal is different.

Outline of the paper. The next section contains an example which illustrates the differences between the compiler of Katz and Yung and our compiler. In Section 3 we present the model in which we reason about security protocols. Section 4 introduces a simple logic and defines security properties within this logic. The protocol transformation is presented in Section 5. Finally, in Section 6 we present our main transfer result and sketch its proof, and then we conclude and give a few directions for future work.

2 Comparison with the Katz and Yung's compiler

Consider the following simple protocol where an agent A sends a session key K_{ab} to B using his public key. Then B acknowledges A's message by forwarding the session key, encrypted under A's public key. We say that this protocol is secure if it preserves the secrecy of K_{ab}.

$$\begin{array}{ll} A \to B: & \{\!|K_{ab}|\!\}_{\mathsf{pk}(B)} \\ B \to A: & \{\!|K_{ab}|\!\}_{\mathsf{pk}(A)} \end{array}$$

Note that this protocol is secure when there is no adversary and is also secure even in the presence of an eavesdropper that may read any message sent over the network but cannot interfere in the protocol.

The resulting protocol obtained after applying Katz and Yung's compiler is the following one.

$$\begin{array}{ll} A \to B: & A, N_a \\ B \to A: & B, N_b \\ A \to B: & [\![\{\!|K_{ab}|\!\}_{\mathsf{pk}(B)}, A, B, N_a, N_b]\!]_{\mathsf{sk}(A)} \\ B \to A: & [\![\{\!|K_{ab}|\!\}_{\mathsf{pk}(A)}, A, B, N_a, N_b]\!]_{\mathsf{sk}(B)} \end{array}$$

However, the compiled protocol is not secure against an adversary that may use corrupted identities. Note that the message $[\![\{\!|K_{ab}|\!\}_{\mathsf{pk}(B)}]\!]_{\mathsf{sk}(A)}$ entirely reveals the message $\{\!|K_{ab}|\!\}_{\mathsf{pk}(B)}$. We assume that the adversary owns a corrupted identity I. The attack works as follows.

$$\begin{array}{llll} (1).1 & A & \to & B: \quad A, N_a \\ (1).2 & B & \to & A: \quad B, N_b \\ (1).3 & A & \to & B: \quad [\![\{\!|K_{ab}|\!\}_{\mathsf{pk}(B)}, A, B, N_a, N_b]\!]_{\mathsf{sk}(A)} \\ (2).1 & I & \to & B: \quad I, N_i \\ (2).2 & B & \to & I: \quad B, N'_b \\ (2).3 & I & \to & B: \quad [\![\{\!|K_{ab}|\!\}_{\mathsf{pk}(B)}, I, B, N_i, N'_b]\!]_{\mathsf{sk}(I)} \\ (2).4 & B & \to & I: \quad [\![\{\!|K_{ab}|\!\}_{\mathsf{pk}(I)}, I, B, N_i, N'_b]\!]_{\mathsf{sk}(B)} \end{array}$$

This allows the intruder to learn any session key used between two honest agents.

In contrast, after applying our own transformation, the resulting protocol would be secure for an unbounded number of sessions, against a fully active attacker.

3 Protocols

In this section we give a language for specifying protocols and define their execution in the presence of passive and active adversaries. For simplicity of presentation, we use a model that does not directly capture probabilistic primitives. Nevertheless, our theorems and proofs easily extend to a model that models randomness explicitly (e.g. through the use of labels as in [6]).

3.1 Syntax

We consider protocols specified in a language similar to the one of [6] allowing parties to exchange messages built from identities and randomly generated nonces using asymmetric and symmetric encryption and digital signatures.

Consider the algebraic signature Σ with the following sorts. A sort ID for agent identities, sorts $\mathsf{SigKey}, \mathsf{VerKey}, \mathsf{AsymEKey}, \mathsf{AsymDKey}, \mathsf{SymKey}$ containing keys for signing, verifying, public-key encryption, public-key decryption, and symmetric encryption algorithms. The algebraic signature also contains sorts Nonce, $\mathsf{Ciphertext}$, $\mathsf{Signature}$, and Pair for nonces, ciphertexts, signatures, and pairs, respectively. The sort Term is a supersort containing, besides all other sorts enumerated above, a sort Int for integers having $\mathbb{Z}$ as the support set. There are eight operations: the four operations $\mathsf{ek}, \mathsf{dk}, \mathsf{sk}, \mathsf{vk}$ are defined on the sort ID and return the asymmetric encryption key, asymmetric decryption key, signing key, and verification key associated to the input identity. The other operations that we consider are pairing, public and symmetric key encryption, and signing. Their ranges and domains are as follows.

- $\langle \_ , \_ \rangle : \mathsf{Term} \times \mathsf{Term} \to \mathsf{Pair}$
- $\{\!|\_|\!\}_\_ : \mathsf{AsymEKey} \times \mathsf{Term} \to \mathsf{Ciphertext}$
- $\{\!\{\_\}\!\}_\_ : \mathsf{SymKey} \times \mathsf{Term} \to \mathsf{Ciphertext}$
- $[\![\_]\!]_\_ : \mathsf{SigKey} \times \mathsf{Term} \to \mathsf{Signature}$

Let $\mathsf{X.a}, \mathsf{X.n}, \mathsf{X.k}, \mathsf{X.c}, \mathsf{X.s}, \mathsf{X.t}$ be sets of variables of sort agent, nonce, (symmetric) key, ciphertext, signature and term respectively, and $\mathsf{X} = \mathsf{X.a} \cup \mathsf{X.n} \cup \mathsf{X.k} \cup \mathsf{X.c} \cup \mathsf{X.s} \cup \mathsf{X.t}$. Protocols are specified using the terms in $T_\Sigma(\mathsf{X})$ of the free algebra generated by X over the signature Σ. We suppose that pairing is left associative and write $\langle m_1, m_2, \ldots, m_l \rangle$ for $\langle\langle\langle m_1, m_2\rangle, m_3\rangle \ldots, m_l\rangle$. When unambiguous, we may omit the brackets.

Throughout the paper we fix a constant $k \in \mathbb{N}$ that represents the number of protocol participants and we write $[k]$ for the set $\{1, 2, \ldots, k\}$. Furthermore, without loss of generality,

we fix the set of agent variables to be $\mathsf{X.a} = \{ A_1, A_2, \ldots, A_k \}$, and partition the set of nonce (and key) variables, according to the party that generates them. Formally:

$$\mathsf{X.n} = \cup_{A \in \mathsf{X.a}} \mathsf{X_n}(A) \text{ where } \mathsf{X_n}(A) = \{N_A^j \mid j \in \mathbb{N}\}$$
$$\mathsf{X.k} = \cup_{A \in \mathsf{X.a}} \mathsf{X_k}(A) \text{ where } \mathsf{X_k}(A) = \{K_A^j \mid j \in \mathbb{N}\}$$

This partition avoids to have to specify later which of the nonces (symmetric keys) are generated by the party executing the protocol, or are expected to be received from other parties.

Roles and protocols. The messages that are sent by participants are specified using terms in$\mathsf{T}_\Sigma(\mathsf{X})$. The individual behavior of each protocol participant is defined by a *role* describing a sequence of message reception/transmission which we call *steps* or *rules*. A k-party protocol consists of k such roles together with an association that maps each step of a role that expects some message m to the step of the role where the message m is produced. Notice that this association essentially defines how the execution of a protocol should proceed in the absence of an adversary.

Definition 1 (Roles and protocols) *The set of* roles *is defined by* $\mathsf{Roles} = \big((\{\,\mathsf{init}\,\} \cup \mathsf{T}_\Sigma(\mathsf{X})) \times (\mathsf{T}_\Sigma(\mathsf{X}) \cup \{\,\mathsf{stop}\,\})\big)^*$. *A* k-party protocol *is a pair* $\Pi = (\mathcal{R}, \mathcal{S})$ *where* $\mathcal{R}$ *is a mapping* $\mathcal{R} : [k] \to \mathsf{Roles}$ *that maps* $i \in [k]$ *to the role executed by the* i*'th protocol participant and* $\mathcal{S} : [k] \times \mathbb{Z} \hookrightarrow [k] \times \mathbb{Z}$ *is a partial mapping that returns for each role-control point pair* (r, p)*, the role-control point pair* $(r', p') = \mathcal{S}(r, p)$ *which emits the message to be processed by role* r *at step* p.

We assume that a protocol specification is such that the r'th role of the protocol $\mathcal{R}(r) = ((\mathsf{rcv}_r^1, \mathsf{snt}_r^1), (\mathsf{rcv}_r^2, \mathsf{snt}_r^2), \ldots)$, is executed by player A_r. Informally, the above definition says that at step p, A_r expects to receive a message of the form specified by rcv_r^p and returns the message snt_r^p. It is important to notice that the terms rcv_r^p and snt_r^p are not actual messages but specify how the message that is received and the message that is output should look like. We note that for technical reasons we sometimes use negative control points (with negatively indexed role rules).

Example 1 *The Needham-Schroeder-Lowe protocol [12]*

$$\begin{aligned} A \to B: &\quad \{\!|N_a, A|\!\}_{\mathsf{ek}(B)} \\ B \to A: &\quad \{\!|N_a, N_b, B|\!\}_{\mathsf{ek}(A)} \\ A \to B: &\quad \{\!|N_b|\!\}_{\mathsf{ek}(B)} \end{aligned}$$

is specified as follows: there are two roles $\mathcal{R}(1)$ *and* $\mathcal{R}(2)$ *corresponding to the sender's role and the receiver's role.*

$$\begin{array}{lll} \mathcal{R}(1): & \big(\mathsf{init},\ \{\!|N_{A_1}^1, A_1|\!\}_{\mathsf{ek}(A_2)}\big) & \mathcal{S}(1,1) = (0,0) \\ & \big(\{\!|N_{A_1}^1, N_{A_2}^1, A_2|\!\}_{\mathsf{ek}(A_1)},\ \{\!|N_{A_2}^1|\!\}_{\mathsf{ek}(A_2)}\big) & \mathcal{S}(1,2) = (2,1) \\ \mathcal{R}(2): & \big(\{\!|N_{A_1}^1, A_1|\!\}_{\mathsf{ek}(A_2)},\ \{\!|N_{A_1}^1, N_{A_2}^1, A_2|\!\}_{\mathsf{ek}(A_1)}\big) & \mathcal{S}(2,1) = (1,1) \\ & \big(\{\!|N_{A_2}^1|\!\}_{\mathsf{ek}(A_2)},\ \mathsf{stop}\big) & \mathcal{S}(2,2) = (1,2) \end{array}$$

Executable protocols. Clearly, not all protocols written using the syntax above are meaningful. We only consider the class of *executable protocols*, *i.e.* protocols for which each role can be implemented in an executable program, using only the local knowledge of the corresponding agent. This requires in particular that any sent message (corresponding to some snt_r^p) is always deducible from the previously received messages (corresponding to $\mathsf{rcv}_r^1, \ldots, \mathsf{rcv}_r^p$). Also we demand that $\mathcal{S}$ is consistent, meaning for example that for a fixed role r, $\mathcal{S}(r,p)$ is defined on exactly $|\mathcal{R}(r)|$ consecutive integers, where $|S|$ denotes the cardinality of the set S. A precise definition of executable protocols may found in the Appendix A.

3.2 Formal Execution Model

We start with the description of the execution model of the protocol in the presence of an active attacker. The model that we consider is rather standard. The parties in the system execute a (potentially unbounded) number of protocol sessions with each other. The communication is under the complete control of the adversary who can intercept, drop, or modify the messages on the network.

The messages transmitted between parties are terms of the algebra T^f freely generated over the signature Σ by an arbitrary fixed set of identities $\mathsf{T}^f_{\mathsf{ID}}$ together with the sets for types SymKey and Nonce defined by:

$$\begin{array}{lcl} \mathsf{T}^f_{\mathsf{SymKey}} & = & \{\mathsf{k}^{a,j,s} \mid a \in \mathsf{T}^f_{\mathsf{ID}}, j \in \mathbb{N}, s \in \mathbb{N}\} \cup \{\mathsf{k}^j \mid j \in \mathbb{N}\} \\ \mathsf{T}^f_{\mathsf{Nonce}} & = & \{\mathsf{n}^{a,j,s} \mid a \in \mathsf{T}^f_{\mathsf{ID}}, j \in \mathbb{N}, s \in \mathbb{N}\} \cup \{\mathsf{n}^j \mid j \in \mathbb{N}\} \end{array}$$

Informally, one should think of the constant $k^{a,j,s}$ (respectively $n^{a,j,s}$) as the j'th key (respectively nonce) generated by party a in session s. Constants k^j and n^j represent keys and nonces produced by the adversary.

To each protocol we associate the set of its valid execution traces. First we clarify what execution traces are and then present the association.

A *global state* of an execution is given by a triple (SId, f, H). Here, SId is the set of role session ids currently executed by protocol participants, f is a global assignment function that keeps track of the local state of each existing session and H is the set of messages that have been sent on the network so far.

More precisely, each session id is a tuple of the form $(s, r, (a_1, a_2, \ldots, a_k))$, where $s \in \mathbb{N}$ is a unique identifier for the session, r is the index of the role that is executed in the session and $a_1, a_2, \ldots, a_k \in \mathsf{T}^f_{\mathsf{ID}}$ are the identities of the parties that are involved in the session. We write SID for the set $(\mathbb{N} \times \mathbb{N} \times (\mathsf{T}^f_{\mathsf{ID}})^k)$ of all session ids.

Mathematically, the global assignment f is a function $f : \mathsf{SId} \to ([\mathsf{X} \hookrightarrow \mathsf{T}^f] \times \mathbb{N} \times \mathbb{Z})$, where $\mathsf{SId} \subseteq \mathsf{SID}$ represents the session ids initialized in the execution. For each such session id $\mathsf{sid} \in \mathsf{SId}$, $f(\mathsf{sid}) = (\sigma, r, p)$ returns its local state. Here, r is the same role index as in sid, the function σ is a substitution, that is a partial instantiation of the variables of the role $\mathcal{R}(r)$ and $p \in \mathbb{Z}$ is the control point of the program. We sometimes write $X\sigma$ for $\sigma(X)$. We denote by GA the set $[\mathsf{SID} \hookrightarrow ([\mathsf{X} \hookrightarrow \mathsf{T}^f] \times \mathbb{N} \times \mathbb{Z})]$ of all possible global assignments.

Finally, the messages that may be sent on the network can be essentially any element of T^f, so we write Msgs for the set 2^{T^f} (where 2^S denotes the power set of S).

An *execution trace* is a sequence

$$(S_0, f_0, H_0) \xrightarrow{\alpha_1} (S_1, f_1, H_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (S_n, f_n, H_n)$$

such that for each $0 \leq i \leq n$, $(S_i, f_i, H_i) \in (2^{\mathsf{SID}} \times \mathsf{GA} \times \mathsf{Msgs})$ and α_i is one of the *actions* **corrupt**, **new**, and **send** with appropriate parameters that we clarify below. This corresponds to the intuition that transitions between two global states are caused by actions of the adversary who can corrupt users, initiate new sessions of the protocol between users that he chooses, and send messages to existing sessions.

For a fixed protocol k-party Π, the transitions between global states are as follows:

- The adversary can corrupt agents: $(\mathsf{SId}, f, H) \xrightarrow{\mathbf{corrupt}(a_1,\ldots,a_l)} (\mathsf{SId}, f, H')$ where $a_1, \ldots, a_l \in \mathsf{T}^f_{\mathsf{ID}}$ and $H' = \cup_{1\leq j\leq l}\mathbf{kn}(a_j) \cup H$. Here, $\mathbf{kn}(a_j)$ denotes the knowledge of a_j: if A is a variable, or a constant of sort agent, we define its knowledge by $\mathbf{kn}(A) = \{\mathsf{dk}(A), \mathsf{sk}(A)\}$ *i.e.* an agent knows its secret decryption and signing key. The adversary corrupts parties by outputting a set of identities. In return, the adversary receives the secret keys corresponding to the identities. In this paper we are only concerned with the case of static corruption so this transition only occurs at the beginning of an execution trace.

- The adversary can initiate new sessions: $(\mathsf{SId}, f, H) \xrightarrow{\mathbf{new}(r,a_1,\ldots,a_k)} (\mathsf{SId}', f', H)$ where $1 \leq r \leq k$, $a_1, \ldots, a_k \in \mathsf{T}^f_{\mathsf{ID}}$, and f' and SId' are defined as follows. Let $s = |\mathsf{SId}| + 1$, be the session identifier of the new session. Then $\mathsf{SId}' = \mathsf{SId} \cup \{(s, r, (a_1, \ldots, a_k))\}$ and the function f' is defined by:

  - $f'(\mathsf{sid}) = f(\mathsf{sid})$ for every $\mathsf{sid} \in \mathsf{SId}$.
  - $f'(s, r, (a_1, \ldots, a_k)) = (\sigma, r, p_0)$ where p_0 is the initial control point of role[1] r and σ is a partial function $\sigma : \mathsf{X} \hookrightarrow \mathsf{T}^f$ defined by

$$\begin{cases} \sigma(A_j) & = a_j & 1 \leq j \leq k \\ \sigma(N^j_{A_r}) & = \mathsf{n}^{a_r,j,s} & j \in \mathbb{N} \\ \sigma(K^j_{A_r}) & = \mathsf{k}^{a_r,j,s} & j \in \mathbb{N} \end{cases}$$

    We recall that the principal that executes role $\mathcal{R}(r)$ is represented by variable A_r thus, in that role, every variable of the form $X^j_{A_r}$ represents a nonce or a symmetric key generated by A_r.

- The adversary can send messages to existing sessions: $(\mathsf{SId}, f, H) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}, f', H')$ where $\mathsf{sid} \in \mathsf{SId}$ and $m \in \mathsf{T}^f$. H' and f' are defined as follows. We define $f'(\mathsf{sid}') =$

[1] The initial control point p_0 is usually 1, but for technical reasons it may also be some other integer.

$\dfrac{}{S \vdash m}\, m \in S$	$\dfrac{}{S \vdash a, \mathsf{ek}(a), \mathsf{vk}(a), \mathsf{k}^j, \mathsf{n}^j}\, j \in \mathbb{N}$	Initial knowledge
$\dfrac{S \vdash m_1 \quad S \vdash m_2}{S \vdash \langle m_1\, , m_2 \rangle}$	$\dfrac{S \vdash \langle m_1\, , m_2 \rangle}{S \vdash m_i}\, i \in \{1,2\}$	Pairing and unpairing
$\dfrac{S \vdash k \quad S \vdash m}{S \vdash \{\!\!\{m\}\!\!\}_k}$	$\dfrac{S \vdash \{\!\!\{m\}\!\!\}_k \quad S \vdash k}{S \vdash m}$	Sym. encryption and decryption
$\dfrac{S \vdash \mathsf{ek}(a) \quad S \vdash m}{S \vdash \{\!\![m]\!\!\}_{\mathsf{ek}(a)}}$	$\dfrac{S \vdash \{\!\![m]\!\!\}_{\mathsf{ek}(a)} \quad S \vdash \mathsf{dk}(a)}{S \vdash m}$	Asym. encryption and decryption
$\dfrac{S \vdash \mathsf{sk}(a) \quad S \vdash m}{S \vdash [\![m]\!]_{\mathsf{sk}(a)}}$	$\dfrac{S \vdash [\![m]\!]_{\mathsf{sk}(a)}}{S \vdash m}$	Signature

Figure 1: Deduction rules for the formal adversary. In the above, $a \in \mathsf{T}_{\mathsf{ID}}^f$.

$f(\mathsf{sid}')$ for every $\mathsf{sid}' \in \mathsf{SId} \setminus \{\mathsf{sid}\}$. Let $f(\mathsf{sid}) = (\sigma, r, p)$ for some σ, r and p, and let $\mathcal{R}(r) = \big((\mathsf{rcv}_r^1, \mathsf{snt}_r^1), \ldots, (\mathsf{rcv}_r^{k_r}, \mathsf{snt}_r^{k_r})\big)$ be the role executed in this session. There are two cases:

- Either there exists a substitution σ' such that $m = \mathsf{rcv}_r^p \sigma'$ and σ' extends σ, that is, $X\sigma' = X\sigma$ whenever σ is defined on X. Then $f'(\mathsf{sid}) = (\sigma', r, p+1)$ and $H' = H \cup \{\mathsf{snt}_r^p \sigma'\}$. We say that m is *accepted*.
- Or we define $f'(\mathsf{sid}) = f(\mathsf{sid})$ and $H' = H$ (the state remains unchanged).

As usual, we are only interested in *valid* execution traces – those traces where the adversary only sends messages that he can compute out of his knowledge and the messages it had seen on the network. The adversary can derive new information using the relation $\vdash$. Intuitively, $S \vdash m$ means that the adversary is able to compute the message m from the set of messages S; the adversarial abilities are captured by the definition in Figure 1. The only rule that is perhaps less standard is the last one. It essentially states that out of a signature an adversary could compute the message that is signed, which is theoretically possible for any secure digital signature scheme.

The set of valid execution traces is formally described by the following definition.

Definition 2 (Valid execution traces) *An execution trace* $(\mathsf{SId}_0, f_0, H_0) \to \ldots \to (\mathsf{SId}_n, f_n, H_n)$ *is* valid *if*

- $H_0 = \mathsf{SId}_0 = \emptyset$, $(\mathsf{SId}_0, f_0, H_0) \rightarrow (\mathsf{SId}_1, f_1, H_1)$ *for one of the three transitions described above and for every* $1 \leq i \leq n$, $(\mathsf{SId}_i, f_i, H_i) \rightarrow (\mathsf{SId}_{i+1}, f_{i+1}, H_{i+1})$ *for one of the last two transitions described above;*
- *moreover, the messages sent by the adversary can be computed using* $\vdash$*, that is whenever* $(\mathsf{SId}_i, f_i, H_i) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}_{i+1}, f_{i+1}, H_{i+1})$ *it holds that* $H_i \vdash m$.

Given a protocol Π*, we write* $\mathsf{Exec}(\Pi)$ *for the set of valid execution traces of* Π.

Example 2 *Playing with the Needham-Schroeder-Lowe protocol described in Example 1, an adversary can corrupt an agent* a_3*, start a new session for the second role with players* a_1, a_2 *and send the message* $\{\!|\mathsf{n}(a_3,1,1), a_1|\!\}_{\mathsf{ek}(a_2)}$ *to the player of the second role. The corresponding valid trace execution is:*

$$(\emptyset, f_1, \emptyset) \xrightarrow{\mathbf{corrupt}(a_3)} (\emptyset, f_1, \mathbf{kn}(a_3)) \xrightarrow{\mathbf{new}(2,a_1,a_2)} (\{\mathsf{sid}_1\}, f_2, \mathbf{kn}(a_3)) \xrightarrow{\mathbf{send}(\mathsf{sid}_1, \{\!|n_3,a_1|\!\}_{\mathsf{ek}(a_2)})} \left(\{\mathsf{sid}_1\}, f_3, \mathbf{kn}(a_3) \cup \{\!|n_3, n_2, a_2|\!\}_{\mathsf{ek}(a_1)}\}\right),$$

where $\mathsf{sid}_1 = (1, 2, (a_1, a_2))$*,* $n_2 = \mathsf{n}(a_2, 1, 1)$*,* $n_3 = \mathsf{n}(a_3, 1, 1)$*, and* f_2, f_3 *are defined as follows:* $f_2(\mathsf{sid}_1) = (\sigma_1, 2, 1)$*,* $f_3(\mathsf{sid}_1) = (\sigma_2, 2, 2)$ *where* $\sigma_1(A_1) = a_1$*,* $\sigma_1(A_2) = a_2$*,* $\sigma_1(N^1_{A_2}) = n_2$*, and* σ_2 *extends* σ_1 *by* $\sigma_2(N^1_{A_1}) = n_3$.

Given an arbitrary trace $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ with $n \in \mathbb{N}$, we define the set of *corrupted agents* of a trace tr by $\mathsf{CA}(\mathsf{tr}) = \{a_1, \ldots, a_l\}$ if $\alpha_1 = \mathbf{corrupt}(a_1, \ldots, a_l)$ and $\mathsf{CA}(\mathsf{tr}) = \emptyset$ otherwise. The set $\mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ of honest session identifiers is the set of session identifiers that correspond to sessions between non-corrupt agents:

$$\mathsf{SId}^{\mathsf{h}}(\mathsf{tr}) = \{\mathsf{sid} \in \mathsf{SId}_n \mid \mathsf{sid} = (s, r, (a_1, \ldots, a_k)), \mathsf{CA}(\mathsf{tr}) \cap \{a_1, \ldots, a_k\} = \emptyset\}.$$

Also, for a trace tr we denote by $\mathsf{I}(\mathsf{tr})$ the set of indexes i of the transitions and global states of tr. For example the above trace has $\mathsf{I}(\mathsf{tr}) = \{0, 1, \ldots, n\}$. If sid is a session id then we denote by $\mathsf{Ag}(\mathsf{sid})$ the set of agents involved in this session, that is $\mathsf{Ag}(\mathsf{sid}) = \{a_1, \ldots, a_k\}$ when $\mathsf{sid} = (\cdot, \cdot, (a_1, \ldots, a_k))$.

4 Security properties

We use a simple logic introduced in [5] to express security properties for protocols specified in the language given in the previous section. We recall this logic, define its semantics and provide several examples of security properties that can be expressed within these logic.

4.1 A logic for security properties

We define the set of local states of a trace $\mathsf{tr} = (\mathsf{SId}_i, f_i, H_i)_{1 \leq i \leq n}$ for role r at step p by

$$\mathcal{LS}_{r,p}(\mathsf{tr}) = \{(\sigma, r, p) \mid \exists i \in [n], \exists \mathsf{sid} \in \mathsf{SId}_i, \text{ s.t. } f_i(\mathsf{sid}) = (\sigma, r, p)\}.$$

We assume an infinite set XSub of meta-variables for substitutions. The logic contains tests between terms where variables are substituted by variable substitutions. More formally, let $\mathsf{T}_{\mathsf{Sub}}$ be the algebra defined by:

$$\mathsf{T}_{\mathsf{Sub}} \quad ::= \quad \varsigma(X) \mid g(\mathsf{T}_{\mathsf{Sub}}) \mid h(\mathsf{T}_{\mathsf{Sub}}, \mathsf{T}_{\mathsf{Sub}})$$

where $\varsigma \in \mathsf{XSub}$, $X \in \mathsf{X}$, and $g, h \in \Sigma$ of arity 1 and 2 respectively.

Besides standard propositional connectors, the logic has a predicate to specify honest agents, equality and inequality tests between terms, and existential and universal quantifiers over the local states of agents.

Definition 3 *The formulas of the logic $\mathcal{L}$ are defined by induction as follows:*

$$\begin{array}{rcl} \phi(\mathsf{tr}) & ::= & \mathsf{NC}(\mathsf{tr}, \varsigma(A)) \mid t_1 = t_2 \mid \neg\phi(\mathsf{tr}) \mid \phi(\mathsf{tr}) \wedge \phi(\mathsf{tr}) \mid \phi(\mathsf{tr}) \vee \phi(\mathsf{tr}) \\ & & \mid \forall \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma\ \phi(\mathsf{tr}) \mid \exists \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma\ \phi(\mathsf{tr}) \mid \exists! \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma\ \phi(\mathsf{tr}) \end{array}$$

where tr is a parameter of the formula, $A \in \mathsf{X}.a$, $\varsigma \in \mathsf{XSub}$, $t_1, t_2 \in \mathsf{T}_{\mathsf{Sub}}$ and $r, p \in \mathbb{N}$. As usual, we may use $\phi_1 \Rightarrow \phi_2$ as a shortcut for $\neg\phi_1 \vee \phi_2$.

Here the predicate $\mathsf{NC}(\mathsf{tr}, \varsigma(A))$ of arity 2 is used to specify non corrupted agents. The quantifications $\forall \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma$ and $\exists \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma$ are over local states of agent r at step p in trace tr, and they bound the variable substitution ς. The semantics of our logic is defined for *closed* formula as follows: standard propositional connectors and negation are interpreted as usual. Equality is syntactic equality. The interpretation of quantifiers and the predicate NC is shown in Figure 2.

A *security property* ϕ should be seen in this paper as an abstraction of the form $\phi \triangleq \lambda\mathsf{tr}.\phi(\mathsf{tr})$, where the tr parameter is used only to define the semantics of such formulas. By abuse of notation we therefore ignore this parameter and write $\phi \in \mathcal{L}$ for a security property. Informally, a protocol Π satisfies ϕ if $\phi(\mathsf{tr})$ is true for all traces of Π. Formally:

Definition 4 (Satisfiability) *Let Π be a protocol and $\phi \in \mathcal{L}$ be a security property. We say that* Π satisfies *the security property ϕ, and write $\Pi \models \phi$ if for any trace $\mathsf{tr} \in \mathsf{Exec}(\Pi)$, $[\![\phi(\mathsf{tr})]\!] = 1$.*

4.2 Examples of security properties

In this section we show how to specify secrecy and several variants of authentication, including those from Lowe's hierarchy [13], in the given security logic.

4.2.1 A secrecy property.

Let Π be a k-party executable protocol. To specify our secrecy property we use a standard encoding. Namely, we add a role to the protocol, $\mathcal{R}(k+1) = (Y, \mathsf{stop})$, where Y is a new variable of sort Term. It can be seen as some sort of witness as it does nothing but waits for receiving a public data.

$$
\begin{array}{rcl}
[\![\mathsf{NC}(\mathsf{tr},a)]\!] & = & \begin{cases} 1 & \text{if } a \text{ does not appear in a corrupt action,} \\ & \textit{i.e.} \text{ if } e_1 \xrightarrow{\mathbf{corrupt}(a_1,\ldots,a_l)} e_2, \text{ where } \mathsf{tr}=(e_1,e_2,\ldots,e_n), \\ & \text{we have } a \neq a_i, \forall 1 \leq i \leq l, \\ 0 & \text{otherwise} \end{cases} \\
[\![\forall \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma\ \phi(\mathsf{tr})]\!] & = & \begin{cases} 1 & \text{if } \forall(\sigma,r,p) \in \mathcal{LS}_{r,p}(\mathsf{tr}), \text{ we have } [\![\phi(\mathsf{tr})[\sigma/\varsigma]]\!]=1, \\ 0 & \text{otherwise.} \end{cases} \\
[\![\exists \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma\ \phi(\mathsf{tr})]\!] & = & \begin{cases} 1 & \text{if } \exists(\sigma,r,p) \in \mathcal{LS}_{r,p}(\mathsf{tr}), \text{ s.t. } [\![\phi(\mathsf{tr})[\sigma/\varsigma]]\!]=1, \\ 0 & \text{otherwise.} \end{cases} \\
[\![\exists! \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma\ \phi(\mathsf{tr})]\!] & = & \begin{cases} 1 & \text{if } \exists! \,\mathsf{sid} \in \mathsf{SId}(\mathsf{tr}), \exists i \in \mathsf{I}(\mathsf{tr}) \text{ s.t.} \\ & f_i(\mathsf{sid})=(\sigma,r,p) \text{ and } [\![\phi(\mathsf{tr})[\sigma/\varsigma]]\!]=1, \\ 0 & \text{otherwise.} \end{cases}
\end{array}
$$

Figure 2: Interpretation of formulas in $\mathcal{L}$.

Informally, the definition of the secrecy property ϕ_s states that, for any local state of an agent playing role r in which a nonce (or a key) X was created in an honest session, a witness (i.e. an agent playing role $k+1$) cannot gain any knowledge on X. Formally, the property is specified by the following formula:

$$\phi_s(\mathsf{tr}) = \forall \mathcal{LS}_{r,1}(\mathsf{tr}).\varsigma \Big(\bigwedge_{l\in[k]} NC(\mathsf{tr},\varsigma(A_l)) \Rightarrow \forall \mathcal{LS}_{k+1,2}(\mathsf{tr}).\varsigma' \left(\varsigma(X) \neq \varsigma'(Y)\right)\Big)$$

As a side remark, notice that we could model also the secrecy of a data X that is received in an honest session: we would simply specify the control point p (instead of 1) at which the date is received by the role r. Moreover, in both cases (that is, X created or received) the formula is always true for honest, single session traces. It will follow that our transformation preserves secrecy of all nonces or keys used in sessions that involve only honest parties.

4.2.2 Authentication properties

We show two formulations for authentication, one for the two-party case, and a second one for the multi-party case.

We show how to use the logic defined above to specify the injective agreement [13] between two parties A and B. Informally, this property states that whenever an A completes a run of the protocol, apparently with B, then there is unique run of B apparently with A such that two agents agree on the values of some fixed variables, provided that A and B are honest. As usual nothing is guaranteed in sessions involving corrupted agents.

Let p_1 be the length of A's role and p_2 be the control point at which B should have received all data items from A. Then, the above intuition is captured by the following formula:

$$\phi_a(\mathsf{tr}) \triangleq \forall \mathcal{LS}_{1,p_1}(\mathsf{tr}).\varsigma \Big(\mathsf{NC}(\mathsf{tr}, A\varsigma) \wedge \mathsf{NC}(\mathsf{tr}, B\varsigma) \Rightarrow \exists! \mathcal{LS}_{2,p_2}(\mathsf{tr}).\varsigma' \left((A\varsigma = A\varsigma') \wedge (B\varsigma = B\varsigma') \wedge \bigwedge_{1\leq i\leq n} (X_i\varsigma = X_i\varsigma')\right)\Big)$$

It is straightforward to modify this formula in order to obtain formulas corresponding to the other variants of authentication from Lowe's hierarchy: to obtain non-injective agreement one should use $\exists$ instead of $\exists!$. To obtain weak agreement one would require equality only on the identities of agents; and finally, to obtain aliveness, one would need only that $(B\varsigma = B\varsigma')$. We give the full specifications of these properties in Appendix D.

4.2.3 Multi-party authentication

We model a simple security property for the multi-party case by requiring that each party authenticates any other party in the sense that each agent is convinced that the other agents were alive in the session. In our logic, this translates to the formula:

$$\phi_{ma}(\mathsf{tr}) = \bigwedge_{i\in[k]} \phi_{ma}(i, \mathsf{tr})$$

with

$$\phi_{ma}(r, \mathsf{tr}) \triangleq \forall \mathcal{LS}_{r,p_r}(\mathsf{tr}).\varsigma \Big(\bigwedge_{l\in[k]} \mathsf{NC}(\mathsf{tr}, A_l\varsigma) \Rightarrow \bigwedge_{i\in[k], i\neq r} \big(\exists \mathcal{LS}_{i,1}(\mathsf{tr}).\varsigma_i \bigwedge_{j\in[k]} (A_j\varsigma = A_j\varsigma_i)\big)\Big)$$

where for each role r p_r is its final control point. We could enforce the property as for the two-party case by changing the set of equalities that should hold.

5 Transformation of protocols

The core idea of the transformation is to have parties agree on some common, dynamically generated, session identifier s, and then transmit the encryption of a message m of the original protocol accompanied by a signature on $m||s$. As discussed in the introduction, the transformation thus ensures that messages sent in a particular session of the protocol can only be accepted by parties engaged in that very same session.

The modification of the source protocol is performed in two steps. We first introduce an initialization phase, where each agent generates a fresh nonce which is distributed to all other participants. The idea is that the concatenation of all these nonces and all the identities involved in the session plays the role of a unique session identifier. To avoid

underspecification of the resulting protocol we fix a particular way in which the nonces are distributed. The resulting first phase of the transformed protocol (which we call Π^{init}) can be informally described as follows:

$$\Pi^{\mathsf{init}}: \quad \begin{cases} A_1 & \to A_2: & N_{A_1} \\ A_2 & \to A_3: & N_{A_1}, N_{A_2} \\ & \vdots & \\ A_{k-1} & \to A_k: & N_{A_1}, N_{A_2} \ldots, N_{A_{k-1}} \\ A_k & \to A_{k-1}: & N_{A_1}, N_{A_2} \ldots, N_{A_{k-1}}, N_{A_k} \\ A_{k-1} & \to A_{k-2}: & N_{A_1}, N_{A_2} \ldots, N_{A_{k-1}}, N_{A_k} \\ & \vdots & \\ A_2 & \to A_1: & N_{A_1}, N_{A_2} \ldots, N_{A_{k-1}}, N_{A_k} \end{cases}$$

We remark that the precise order in which participants send these nonces does not really matter, and we do not require that these nonces be authenticated in some way. In principle an active adversary is allowed to forward, block or modify the messages sent during the initialization phase, but behaviors that deviate from the intended execution of the protocol are detected in the next phase.

In the second phase of the transformed protocol, the execution proceeds as prescribed by Π with the difference that to each message m that needs to be sent, the sending parties also attaches a signature $[\![m, p, \mathsf{nonces}]\!]_{\mathsf{sk}'(a)}$ and encrypts the whole construct with the intended receiver public key. p is the current control point and nonces is the concatenation of the nonces received during the first phase with the identities of the participants involved in the protocol. To avoid confusion and unintended interactions between the signatures and the encryptions produced by the compiler and those used in the normal execution of the protocol, the former use fresh signatures and public keys. Formally, we extend the signature Σ with four new function symbols sk', vk', ek' and dk' which have exactly the same functionality (that is the same sort and similar deduction rules) with sk, vk, ek and dk respectively. This formalises the assumption that in the transformed version of Π each agent a has associated two pairs of verification/signing keys $((\mathsf{vk}(a), \mathsf{sk}(a))$ and $(\mathsf{vk}'(a), \mathsf{sk}'(a)))$ and two pairs of encryption/decryption keys $((\mathsf{ek}(a), \mathsf{dk}(a))$ and $(\mathsf{ek}'(a), \mathsf{dk}'(a)))$ and that these new pairs of keys were correctly distributed previously to any execution of the protocol. We assume that source protocols are constructed over Σ only.

Definition 5 (Transformed protocol) *Let $\Pi = (\mathcal{R}, \mathcal{S})$ be a k-party executable protocol such that the nonce variables $N^0_{A_k}$ do not appear in $\mathcal{R}$ (which can be ensured by renaming the nonce variables of Π) and all the initial control points are set to 1 (which can be ensured by rewriting the function $\mathcal{S}$).*

The transformed protocol $\widetilde{\Pi} = (\widetilde{\mathcal{R}}, \widetilde{\mathcal{S}})$ *is defined as follows: $\widetilde{\mathcal{R}}(r) = \mathcal{R}^{\mathsf{init}}(r) \cdot \mathcal{R}'(r)$ and $\widetilde{\mathcal{S}} = \mathcal{S}^{\mathsf{init}} \cup \mathcal{S}$ where $\cdot$ denotes the concatenation of sequences and $\mathcal{R}^{\mathsf{init}}$, $\mathcal{R}'$ and $\mathcal{S}^{\mathsf{init}}$ are*

defined as follows:

$$\mathcal{R}^{\mathsf{init}}(r) = \big((\mathsf{nonces}_{r-1}, \mathsf{nonces}_r), (\mathsf{nonces}_k, \mathsf{nonces}_k)\big), \quad \forall 1 \leq r < k,$$
$$\mathcal{S}^{\mathsf{init}}(r,-1) = (r-1,-1), \quad \mathcal{S}^{\mathsf{init}}(r,0) = (r+1,0), \quad \forall 1 \leq r < k,$$
$$\mathcal{R}^{\mathsf{init}}(k) = \big((\mathsf{nonces}_{k-1}, \mathsf{nonces}_k)\big)$$
$$\mathcal{S}^{\mathsf{init}}(k,0) = (k-1,-1)$$

with $\mathsf{nonces}_0 = \mathsf{init}$ *and* $\mathsf{nonces}_j = \langle N^0_{A_1}, N^0_{A_2}, \ldots, N^0_{A_j}\rangle$ *for* $1 \leq j \leq k$.
Let $\mathcal{R}(r) = \big((\mathsf{rcv}^p_r, \mathsf{snt}^p_r)\big)_{p\in[k_r]}$. *Then* $\mathcal{R}'(r) = \big((\widetilde{\mathsf{rcv}}^p_r, \widetilde{\mathsf{snt}}^p_r)\big)_{p\in[k_r]}$ *such that if* $\mathsf{rcv}^p_r = \mathsf{init}$ *then* $\widetilde{\mathsf{rcv}}^p_r = \mathsf{fake}$, *if* $\mathsf{snt}^p_r = \mathsf{stop}$ *then* $\widetilde{\mathsf{snt}}^p_r = \mathsf{stop}$ *and otherwise*

$$\widetilde{\mathsf{rcv}}^p_r = \{\![\mathsf{rcv}^p_r, [\![\mathsf{rcv}^p_r, p', \mathsf{nonces}]\!]_{\mathsf{sk}'(A_{r'})}]\!\}_{\mathsf{ek}'(A_r)},$$
$$\widetilde{\mathsf{snt}}^p_r = \{\![\mathsf{snt}^p_r, [\![\mathsf{snt}^p_r, p, \mathsf{nonces}]\!]_{\mathsf{sk}'(A_r)}]\!\}_{\mathsf{ek}'(A_{r''})}$$

where $(r', p') = \mathcal{S}(r, p)$, $(r, p) = \mathcal{S}(r'', p'')$ *and* $\mathsf{nonces} = \langle A_1, \ldots, A_k, \mathsf{nonces}_k\rangle$.

The initial control point is now set to -1 (or 0 for A_k) since actions have been added for the initialization stage. This avoids shifting the control points and clarifies the relation between actions in the original and transformed protocol.

The special message fake is used to model a *fake* reception or transmission. We use this message to model the situation where an agent waits for more than one message in order to reply or when, in response to some message, an agent sends more then one reply: so (fake, m) and (m, fake) do not represent reception/transmission steps but only a transmission step or respectively a reception step.

6 Main result

In this section we give our main result. We first formalise honest, single session executions for protocols. Then we identify a fragment of the logic defined in Section 4 whose formulas are transferred through our transformation, and then prove our transference theorem.

6.0.4 Honest, single session traces

We identify a class of executions, which we call honest, single session executions which, intuitively, correspond to traces where just one session is executed, session in which all parties are honest and the adversary did not interfere.

Definition 6 (Honest, single session trace) *Let* $\Pi = (\mathcal{R}, \mathcal{S})$ *be a* k*-party protocol and* $\mathsf{tr} = (\mathsf{Sld}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{Sld}_n, f_n, H_n)$ *be a valid trace of* Π. *The trace* tr *is an* honest, single session trace *if there are* k *agent identities* $a_1, \ldots, a_k$ *such that*

- *for* $1 \leq i \leq k$, $\alpha_i = \mathbf{new}(i, a_1, \ldots, a_k)$,

- *for* $k+1 \leq i \leq n$, $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$, $m = \mathsf{rcv}_r^p\sigma$ *where* $f_i(\mathsf{sid}) = (\sigma, r, p+1)$, *and there exists* $j < i$ *such that* $f_j(\mathsf{sid}') = (\sigma', r', p')$, $\mathcal{S}(r,p) = (r', p')$, *and* $m = \mathsf{snt}_{r'}^{p'}\sigma'$ *for some* sid'.

Let $\mathsf{Exec}^{p,1}(\Pi)$ be the set of honest, single session traces of Π.

Definition 7 (Passive, single session satisfiability) *Let* Π *be a protocol and* $\phi \in \mathcal{L}$ *be a security property. We say that* Π *satisfies the security property* ϕ *for* passive adversaries and a single session*, and write* $\Pi \models^{p,1} \phi$ *if for any trace* $\mathsf{tr} \in \mathsf{Exec}^{p,1}(\Pi)$, $[\![\phi(\mathsf{tr})]\!] = 1$.

6.0.5 Transferable security properties

We now define a fragment $\mathcal{L}'$ of $\mathcal{L}$ for which the transference result outlined above holds.

Definition 8 *The set* $\mathcal{L}'$ *consists of those formulas* $\phi(\mathsf{tr})$ *with*

$$\phi(\mathsf{tr}) = \forall \mathcal{LS}_{r,p}(\mathsf{tr}).\varsigma \Big(\bigwedge_{l\in[k]} \mathsf{NC}(\mathsf{tr}, \varsigma(A_l)) \Rightarrow \bigwedge_{i\in I} \big(\mathcal{Q}_i \, \mathcal{LS}_{r_i,p_i}(\mathsf{tr}).\varsigma_i \bigwedge_{j\in J_i} \tau_j^i(u_j^i, v_j^i) \big) \Big)$$

where $\mathcal{Q}_i \in \{\forall, \exists, \exists!\}$, *and for all* $i \in I$, *for all* $j \in J_i$, *if* $\mathcal{Q}_i = \forall$ *then* $\tau_j^i \in \{\neq\}$ *and if* $\mathcal{Q}_i \in \{\exists, \exists!\}$ *then* $\tau_j^i \in \{=, \neq\}$*; moreover, for each* $i \in I$, *if* $\mathcal{Q}_i = \forall$ *(respectively* $\mathcal{Q}_i = \exists!$*) then for all (there is)* $j \in J_i$ *we have that (such that* $\tau_j^i \in \{=\}$ *and) there exists at least a subterm* $\varsigma(X)$ *in* u_j^i *or* v_j^i *with* X *a nonce or key variable.*

As usual, we require security properties to hold in sessions between honest agents. This means that no guarantee is provided in a session where a corrupted agent is involved. But this does not prevent honest agents from contacting corrupted agents in parallel sessions. Properties that can be expressed in our fragment $\mathcal{L}'$ are correspondence relations between (data in) particular local states of agents in different sessions. It is a non-trivial class since e.g. the logical formulas given in Section 4 for expressing secrecy and authentication are captured by the above definition.

6.0.6 Transference result

The main result of this paper is the following transference theorem. It informally states that the formulas of $\mathcal{L}'$ that are satisfied in single, honest executions of a protocol are also satisfied by executions of the transformed protocol in the presence of a fully active adversary.

Theorem 1 *Let* Π *be a protocol and* $\widetilde{\Pi}$ *the corresponding transformed protocol. Let* $\phi \in \mathcal{L}'$ *be a security property. Then*

$$\Pi \models^{p,1} \phi \;\Rightarrow\; \widetilde{\Pi} \models \phi.$$

The main intuition behind the proof is a property that our transformation enjoys, namely that any execution in the presence of an active adversary is closely mirrored by some *honest* execution (i.e. an execution with no adversarial interference plus some additional useless sessions). We define honest executions next.

6.0.7 Honest executions

Recall that we demand that protocols come with an intended execution order, in which the designer specifies the source of each message in an execution. Roughly, in an honest execution trace one can partition the set of session ids in sets of at most k role sessions (each corresponding to a different role of the protocol) such that messages are exchanged only within partitions, and the message transmission within each partition follows the intended execution specification. Since we cannot prevent an intruder to create new messages and sign them with corrupted signing keys, clearly the property can hold only for session identifiers corresponding to honest participants. The above ideas are captured by the following definition.

Definition 9 (Honest execution traces) *Let Π be an executable protocol. An execution trace $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ is* honest *if it is valid and there is a partition PrtSId of the honest role session identifiers $\mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ such that:*

1. *for all $S \in \mathsf{PrtSId}$, for all $\mathsf{sid}, \mathsf{sid}' \in S$ with $\mathsf{sid} \neq \mathsf{sid}'$ and $\mathsf{sid} = (s, r, (a_1, \ldots a_k))$ and $\mathsf{sid}' = (s', r', (a'_1, \ldots a'_k))$, we have $r \neq r'$, and $a_j = a'_j$ for all $1 \leq j \leq k$; that is, in any protocol session each of the participants execute different roles[2] and the agents agree on their communication partners;*

2. *whenever $(\mathsf{SId}_{i-1}, f_{i-1}, H_{i-1}) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}_i, f_i, H_i)$ with $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$, m accepted, $m \neq \mathsf{fake}$ and $m = \mathsf{rcv}^p_r\sigma$, $p \geq 1$, we have that there are $\mathsf{sid}' \in [\mathsf{sid}]$ and $i' < i$ such that $r' = r''$, $p' = p''$ and $m = \mathsf{snt}^{p'}_{r''}\sigma'$ where $f_i(\mathsf{sid}) = (\sigma, r, p+1)$, $f_{i'}(\mathsf{sid}') = (\sigma', r', p')$, $\mathcal{S}(r, p) = (r'', p'')$ and $[\mathsf{sid}]$ denotes the partition to which sid belongs to.*

We write $\mathsf{Exec}^{\mathsf{h}}(\Pi)$ for the set of honest execution traces of Π.

Notice that the above definition considers partial executions in which not all roles finish their execution, and where not all roles in a protocol session need to be initialized.

The following lemma states that for any transformed protocol, an active intruder cannot interfere with the execution of honest sessions and honest sessions share the same value for $\mathsf{sessionID}$. Here, interference means that the adversary is capable to construct and send some message (not produced by some party in the protocol) to a party, such that the party accepts that message. Notice that we are not concerned with the possibility that the adversary may not deliver (and therefore block) messages of a session.

Lemma 1 *Let Π be a protocol and $\widetilde{\Pi}$ the corresponding transformed protocol. In $\widetilde{\Pi}$, any valid execution trace is an honest execution trace.*

Proof sketch: Let tr be a valid execution trace of $\widetilde{\Pi}$. We construct the partition of session ids by simply grouping session ids that have the same value of nonces; we write $\mathsf{PrtSId}(\mathsf{tr})$ for the resulting partition. It is easy to check that $\mathsf{PrtSId}(\mathsf{tr})$ satisfies the first condition

[2] Consequently, each partition consists of at most k role sessions.

of the definition of an honest execution trace. We prove that the second condition also holds by induction on the length of the trace. Assume that $(\mathsf{SId}_{i-1}, f_{i-1}, H_{i-1}) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}_i, f_i, H_i)$ with $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$, m accepted, $m \neq \mathsf{fake}$ and $m = \widetilde{\mathsf{rcv}}^p_r\sigma$, $p \geq 1$. Then, m must be of the form $\{\![m', [\![m', p, m_0]\!]_{\mathsf{sk}'(a)}]\!\}_{\mathsf{ek}'(b)}$ with a, b honest agents and the agents occurring in m_0 being honest too. Since the adversary cannot forge $[\![m', p, m_0]\!]_{\mathsf{sk}'(a)}$, a message of the form $\{\![m', [\![m', p, m_0]\!]_{\mathsf{sk}'(a)}]\!\}_{\mathsf{ek}'(b')}$ must have been sent by the honest agent a in a session $\mathsf{sid}' \in [\mathsf{sid}]$ since the two agents agree on m_0. Thanks to the control point that is also signed, we can show that a must have sent his message exactly to the agent that is expected, and then deduce that $b = b'$ and a's action also satisfies the condition on function $\mathcal{S}$. □

Any nonce or key generated in an honest session is always protected by at least one encryption with a public key of a non-corrupted agent.

Lemma 2 *Let Π be a k-party protocol and $\widetilde{\Pi}$ be the corresponding transformed protocol. Let X be a nonce (or a key) variable of Π, tr be a valid execution trace of $\widetilde{\Pi}$, (SId, f, H) be a global state of tr and t be a message deducible from H, i.e. $H \vdash t$.*

For any honest session id $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ with $f(\mathsf{sid}) = (\sigma, \cdot, \cdot)$, for any occurrence of $X\sigma$ in t (i.e. for any path q such that $t|_q = X\sigma$), $X\sigma$ occurs in messages of the form $m \stackrel{\mathsf{def}}{=} \{\![m', [\![m', p, \sigma(\mathsf{nonces})]\!]_{\mathsf{sk}(a)}]\!\}_{\mathsf{ek}'(b)}$, where b is an honest agent, i.e. $b \notin CA(\mathsf{tr})$.

Proof sketch: Using Lemma 1, we can show that the only possible values for $X\sigma$ are nonces (or keys) generated in honest sessions. Thus $X\sigma$ is initially protected with an honest public key encryption. Then the only way for an adversary to remove that encryption is to send the message to an honest agent, which in turn will send it to one of the agents occurring in $\sigma(\mathsf{nonces})$ thus to another honest agent; still, $X\sigma$ will be protected by an honest public key encryption. □

The next lemma says that every honest protocol session in the transformed protocol executes exactly like an honest protocol session in the initial protocol without intruder interference. To state this formally we need some auxiliary definitions first.

Since for a session in the transformed protocol there are more actions (corresponding to the initial phase), we define the actions we are interested in.

Let Π be an executable protocol. For an honest trace $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ and a partition $[\mathsf{sid}]$ where $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ we define $\mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}])$ to be the set of indexes i such that:

- $\alpha_i = \mathbf{new}(r, a_1, \ldots, a_k)$, where $\mathsf{sid} = (\cdot, \cdot, (a_1, \ldots, a_k))$, or
- $\alpha_i = \mathbf{send}(\mathsf{sid}', m)$ and $\mathsf{sid}' \in [\mathsf{sid}]$, m accepted, and $m = \mathsf{rcv}^p_r\sigma$, $p \geq 1$, where $f_i(\mathsf{sid}') = (\sigma, r, p+1)$.

Note that the definition of $\mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}])$ does not depend on the representant sid.

Lemma 3 *Let* Π *be a protocol and* $\widetilde{\Pi}$ *the corresponding transformed protocol. Then* $\forall \mathsf{tr} \in \mathsf{Exec}^{\mathsf{h}}(\widetilde{\Pi}), \forall \mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$*, there exist* $\mathsf{tr}_0 \in \mathsf{Exec}^{p,1}(\Pi)$*,* $\mathsf{sid}_0 \in \mathsf{SId}(\mathsf{tr}_0)$ *and bijections* $\mathcal{I} : \mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}]) \to \mathsf{Ix}(\mathsf{tr}_0, [\mathsf{sid}_0])$*,* $g : \mathsf{Ag}(\mathsf{sid}) \to \mathsf{Ag}(\mathsf{sid}_0)$ *and* $\varphi : [\mathsf{sid}] \to [\mathsf{sid}_0]$ *such that* $\forall \mathsf{sid}' \in [\mathsf{sid}], \forall i \in \mathsf{Ix}(\mathsf{tr})$*,* $f^0_{\mathcal{I}(i)}(\varphi(\mathsf{sid}')) = (\sigma_0, r, p)$ *with* $\sigma = \sigma_0 \circ g$ *where* $f_i(\mathsf{sid}') = (\sigma, r, p)$*,* $\mathsf{tr} = ((f_i, \cdot, \cdot))_i$ *and* $\mathsf{tr}_0 = ((f^0_j, \cdot, \cdot))_j$*. Moreover, for these* tr_0*,* sid_0 *and bijections the converse also holds, that is,* $\forall \mathsf{sid}'_0 \in \mathsf{SId}(\mathsf{tr}_0), \forall i_0 \in \mathsf{Ix}(\mathsf{tr}_0)$*,* $f_{\mathcal{I}^{-1}(i_0)}(\varphi^{-1}(\mathsf{sid}'_0)) = (\sigma, r, p)$ *with* $\sigma = \sigma_0 \circ g$ *where* $f^0_{i_0}(\mathsf{sid}'_0) = (\sigma_0, r, p)$*.*

The proof of this lemma consists of a simple rewriting of the definition of honest traces into the definition of honest, single session traces.

6.0.8 Sketch of proof of the main result

Firstly, Lemma 1 says that it sufficient to look at honest execution traces in the transformed protocol. So we fix an arbitrary honest trace tr. Then Lemma 3 says that every (eventually partial) honest protocol session in tr can be projected to a (partial) honest, single session trace tr_0 in the initial protocol. Observe that we are only interested in honest sessions of tr, since that is what the hypothesis of the implication in ϕ refers to (i.e. $\bigwedge_{l \in [k]} NC(\mathsf{tr}, A_l\varsigma)$). But then the hypothesis of the implication in ϕ is trivially satisfied for passive, single sessions in Π. Hence also the conclusion of the implication in ϕ is satisfied for passive, single sessions in Π.

We consider three cases according to what the quantifier $\mathcal{Q}_i$ is.

If it is $\exists$ then the there is a local state satisfying the (in)equalities simply because there is one in the honest, single session trace tr_0 (we apply again Lemma 3 to obtain the local states in tr from those in tr_0).

If $\mathcal{Q}_i = \exists!$ we also need to prove uniqueness. From the conditions of ϕ we know that there is $j \in J_i$ such that $\tau^i_j \in \{=\}$ and in (say) u^i_j there is an occurrence of $X\varsigma$ with X a nonce or a key variable. Consider an arbitrary "valid" σ (meaning that σ is a substitution such that $(\sigma, r, p) \in \mathcal{LS}_{r,p}(\mathsf{tr})$). From the previous case we know that there is a valid σ' such that $(u^i_j = v^i_j)[{}^{\sigma}/_{\varsigma}][{}^{\sigma'}/_{\varsigma_i}]$. Suppose that there is also a valid σ'' such that $(u^i_j = v^i_j)[{}^{\sigma}/_{\varsigma}][{}^{\sigma''}/_{\varsigma_i}]$. But then $X\sigma$ occurs both in $Y\sigma'$ and $Y\sigma''$, where Y is a variable of v^i_j such that $X\varsigma$ occurs in v^i_j under $Y\varsigma_i$. Since $Y\sigma'$ and $Y\sigma''$ are parts of sent or received messages, it follows that they are parts of messages known by the intruder and hence $X\sigma$ is also part of a message known by the intruder. Thus we can apply Lemma 2 and obtain that $Y\sigma'$ and $Y\sigma''$ were sent or received in the same honest protocol session as $X\sigma$. And since σ' and σ'' are substitutions obtained at the same control point p_i of the same role r_i it follows that $\sigma' = \sigma''$ hence uniqueness.

Finally consider $\mathcal{Q}_i = \forall$. This case proceeds similarly. If, by absurd, there are local states such that the terms in some test become equal for some σ' then it must be the case that the corresponding session id is honest (this is again obtained using the uniqueness of nonces created in honest sessions, that is Lemma 2). We can then project this equality in the honest, single session trace tr_0 to obtain a contradiction.

7 Conclusions and future work

We have presented a general transformation for security protocols that essentially prevents an active adversary to interfere with the executions of the protocol that involves only honest parties. An important consequence of our transformation is that it enables a transference theorem of a non-trivial class of security properties from a setting where no adversary is present to a setting where a fully active adversary may tamper with the protocol execution. The security properties that are transferred include secrecy and various formulations of authentication.

One interesting avenue for future research it to obtain more general transference theorems between the properties of the original protocol and those of the transformed protocol. It would be also interesting to better develop the modular development approach implied by our results. In particular, it would be interesting to develop a language for building "naive" specification which can then be compiled into secure protocols using our transformation.

Finally, from an efficiency perspective, it is important to look for simpler transformations that make lighter use of cryptographic primitives, perhaps at the expense of ensuring weaker security guarantees for the resulting protocol. We note that the Katz and Yung compiler [11] is one example of such a transformation which merits further investigation.

References

[1] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. *Inf. Comput.*, 174(1):37–83, 2002.

[2] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428, New York, NY, USA, 1998. ACM Press.

[3] B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In A. Gordon, editor, *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, April 2003.

[4] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 148–164. Springer-Verlag, June 2003.

[5] V. Cortier, H. Hördegen, and B. Warinschi. Explicit Randomness is not Necessary when Modeling Probabilistic Encryption. In *Workshop on Information and Computer Security (ICS 2006)*, Timisoara, Romania, September 2006.

[6] V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume

3444 of *Lecture Notes in Computer Science*, pages 157–171, Edinburgh, U.K, April 2005. Springer.

[7] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *J. Comput. Secur.*, 13(3):423–482, 2005.

[8] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.

[9] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[10] L. Gong and P. Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, pages 44–55, 1995.

[11] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proceedings of Crypto'03*, pages 110–125. Springer-Verlag, 2003.

[12] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Margaria and Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes on Computer Science*, pages 147–166, 1996.

[13] G. Lowe. A hierarchy of authentication specifications. In *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97)*, Rockport, Massachusetts, USA, 1997. IEEE Computer Society Press.

[14] G. Lowe. Towards a completeness result for model checking of security protocols. In *Proc. of the 11th Computer Security Foundations Workshop (CSFW'98)*. IEEE Computer Society Press, 1998.

[15] R. Ramanujam and S. P. Suresh. A decidable subclass of unbounded security protocols. In *Proc. IFIP Workshop on Issues in the Theory of Security (WITS'03)*, pages 11–20, Warsaw (Poland), 2003.

[16] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Computer Society Press, 2001.

A Formal Definition of Executable Protocols

In order to define executable protocols, we use another deduction relation $\vdash_p$ which is similar with $\vdash$ but which applies on terms of the free algebra $\mathsf{T}_\Sigma(\mathsf{X})$ (rather then T^f). The only differences are that now $a \in \mathsf{X.a}$ and that the second rule becomes $\dfrac{}{S \vdash a, \mathsf{ek}(a), \mathsf{vk}(a)}$.

Definition 10 *A protocol* $\Pi = (\mathcal{R}, \mathcal{S})$ *with* $\mathcal{R}(r) = ((\mathsf{rcv}_r^1, \mathsf{snt}_r^1), \ldots, (\mathsf{rcv}_r^{k_r}, \mathsf{snt}_r^{k_r})$ *is executable if:*

1. *The protocol has the* executable decryption *property, i.e. for all* $A_r \in \mathsf{X.a}$ *the only encryption keys that are contained in terms* rcv_r^p *(for* $p \in [k_r]$*) are* $\mathsf{ek}(A_r)$*;*
2. *For all* $r \in [k]$*, all* $p \in [k_r]$*, and all* $A_r \in \mathsf{X.a}$ *we require that whenever* rcv_r^p *contains a signature* $[\![t]\!]_{\mathsf{sk}(A)}$ *for some term* $t \in \mathsf{T}_\Sigma(\mathsf{X})$*, the term* t *can be computed from* $\mathsf{rcv}_r^1, \mathsf{rcv}_r^2, \ldots, \mathsf{rcv}_r^p, \mathbf{kn}(A_r)$ *by Dolev-Yao operations, i.e.* $\mathsf{rcv}_r^1, \mathsf{rcv}_r^2, \ldots, \mathsf{rcv}_r^p, \mathbf{kn}(A_r) \vdash_p t$*;*
3. *The messages that are sent are computable: for all* $p \in [k_r]$ *we require that* snt_r^p *can be computed from* $\mathsf{rcv}_r^1, \mathsf{rcv}_r^2, \ldots, \mathsf{rcv}_r^p, \mathbf{kn}(A_r)$ *by Dolev-Yao operations, i.e.* $\mathsf{rcv}_r^1, \mathsf{rcv}_r^2, \ldots, \mathsf{rcv}_r^p, \mathbf{kn}(A_r) \vdash_p \mathsf{snt}_r^p$;
4. *For all* $r \in [k]$*, all* $p \in [k_r]$*, the variables of* snt_r^p *are contained in the union of the variables of* $\mathsf{rcv}_r^1, \ldots, \mathsf{rcv}_r^p$*,* $\mathsf{X.a}$*, and* $\mathsf{X_n}(A_r)$*;*
5. *the function* $\mathcal{S}$ *is injective;*
6. *for each* $r \in [k]$*, there is* $p_0 \in \mathbb{Z}$ *such that if* $p_0 \leq p < p_0 + k_r$ *then* $\mathcal{S}$ *is defined on* (r, p) *and otherwise it is undefined on* (r, p)*.*

The Needham-Schroeder-Lowe protocol, described in Example 1, is executable.

B Proofs of lemmas

B.1 Proof of Lemma 1

The following lemma, called the locality lemma, states a well-known property of the intruder deduction systems and will be used in the proof that follows.

Lemma 4 *Let* S *be a set of terms,* u *a term and* π *a minimal proof of* $S \vdash u$*. Then all terms occurring in the nodes of* π *are in* $St(S, u)$*. If the last rule of* π *is a decomposition then these terms are in* $St(S)$*.*

Let $\Pi = (\mathcal{R}, \mathcal{S})$ be an executable protocol such that there are no variable nonces N_A^0 in $\mathcal{R}$ and the initial control points are set to 1. Let $\mathcal{R}(r) = ((\mathsf{rcv}_r^p, \mathsf{snt}_r^p))_{1 \leq p \leq k_r}$. Consider

an arbitrary valid execution trace tr of the protocol $\widetilde{\Pi}$. Let $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$. We need to show that tr is an honest execution trace.

We first give a few useful definitions and properties.

For a message m we define $\overline{m} = m_0$ if $m = \{\!|m', [\![m', p, m_0]\!]_{\mathsf{sk}'(a)}|\!\}_{\mathsf{ek}'(b)}$ for some identities a, b, some messages m', m_0 and some $p \in \mathbb{Z}$ and $\overline{m} = \bot$ otherwise[3]. We call m_0 the *nonces field*[4] of m.

For any transition $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ such that m is accepted we have that $m = \widetilde{\mathsf{rcv}}_r^p \sigma$ where $f_i(\mathsf{sid}) = (\sigma, r, p+1)$. Suppose that $p \geq 1$ and $\mathsf{rcv}_r^p \neq \mathsf{init}$. Then, since $\overline{\widetilde{\mathsf{rcv}}_r^p} = \mathsf{nonces}$, we have that $\overline{m} = \sigma(\mathsf{nonces})$. The converse also holds, that is if $\overline{m}$ is defined then $p \geq 1$ and $\mathsf{rcv}_r^p \neq \mathsf{init}$.

The following property says that messages sent and accepted in the same role sessions have the same nonces field: If $\mathsf{sid} \in \mathsf{SId}_n$ and $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha_{i'} = \mathbf{send}(\mathsf{sid}, m')$ are two transitions in tr such that m and m' are accepted then m and m' have the same nonces field, provided it is defined for both messages. Indeed the nonces field is given by the substitution in $f_i(\mathsf{sid})$ and $f_{i'}(\mathsf{sid})$ respectively (as we have seen in the previous paragraph). And these substitution are the same on $\mathsf{X.a} \cup \mathsf{X.n}$ since they extend the substitution in $f_{i_0}(\mathsf{sid})$ with $i_0 < \min(i, i')$ where i_0 is such that $\mathsf{SId}_{i_0} \setminus \mathsf{SId}_{i_0-1} = \{\, \mathsf{sid} \,\}$ (that is, α_{i_0} is the **new** transition produced when the session sid was initiated).

Next, we define the relation $\sim$ between role sessions. Intuitively this relation should capture the notion of *protocol session*. That is, two role sessions $(\cdot, r, \cdot)$ and $(\cdot, r', \cdot)$ should be in relation $\sim$ if and only if the two agents playing roles r and r' are communicating in the same protocol session.

We say that two sessions ids $\mathsf{sid}, \mathsf{sid}' \in \mathsf{SId}_n$ are in relation $\sim$ if *there are* two (not necessarily different) transitions in tr labelled by $\alpha = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha' = \mathbf{send}(\mathsf{sid}', m')$ such that m and m' are accepted, $\overline{m} = \overline{m'}$ and $\overline{m} \neq \bot$, that is nonces is instantiated by the same term in the two messages m and m'.

This is relation says in fact more about two role sessions: If $\mathsf{sid} \sim \mathsf{sid}'$ then *for any* two transitions in tr labelled by $\alpha = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha' = \mathbf{send}(\mathsf{sid}', m')$ such that m and m' are accepted and $\bot \notin \{\overline{m}, \overline{m'}\}$, we have that $\overline{m} = \overline{m'}$. This is easy to verify using the above stated property (that is, messages sent and accepted in the same role sessions have the same nonces field). A(nother) direct consequence is that if in a session sid the agent executing this session started the second phase (that is he received a valid message m with $\overline{m} \neq \bot$) then $\mathsf{sid} \sim \mathsf{sid}$.

But there may be sessions in which agents are still in the initialization phase. In these sessions the messages m sent so far have no nonces field and thus the relation $\sim$ doesn't capture them (it is not "defined" on these sessions). However we are not interested in these (role) sessions and so we do not group them into protocol sessions. But technically we need a

[3] Hence $\overline{\cdot}$ is a partial function from terms to terms and $\bot$ means undefined.

[4] The definition of $\widetilde{\mathsf{rcv}}_r^p$ for $p \geq 1$ and the following paragraph provide an explanation for choosing this name. Recall that $\mathsf{nonces} = \langle A_1, \ldots, A_k, N^0_{A_1}, N^0_{A_2}, \ldots, N^0_{A_k} \rangle$ and $\widetilde{\mathsf{rcv}}_r^p = \{\!|\mathsf{rcv}_r^p, [\![\mathsf{rcv}_r^p, p', \mathsf{nonces}]\!]_{\mathsf{sk}'(A_{r'})}|\!\}_{\mathsf{ek}'(A_r)}$.

partition, hence we simply consider the reflexive closure of $\sim$, denoted $\sim'$. This means that those $\mathsf{sid} \in \mathsf{SId}_n$ for which there is no transition labelled by $\mathbf{send}(\mathsf{sid}, m)$, with m accepted and $\overline{m} \neq \bot$, are only in relation with themselves. The relation $\sim'$ is clearly an equivalence relation. We consider PrtSId to by the quotient set of $\mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ by $\sim'$.

We prove next that the partition PrtSId satisfies the conditions in the definition of honest executions.

Let us look at the first point of Definition 9. Consider two arbitrary session ids $\mathsf{sid}, \mathsf{sid}' \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ such that $\mathsf{sid} \sim' \mathsf{sid}'$ and $\mathsf{sid}' \neq \mathsf{sid}$. Let $\mathsf{sid} = (s, r, (a_1, \ldots, a_k))$ and $\mathsf{sid} = (s', r', (a'_1, \ldots, a'_k))$. By the definition of $\sim$ we have that there are two transitions $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha_{i'} = \mathbf{send}(\mathsf{sid}', m')$ such that m and m' have the same nonces field (besides other things). Let $f_i(\mathsf{sid}) = (\sigma, r, p+1)$ and $f_{i'}(\mathsf{sid}') = (\sigma', r', p'+1)$. We have that $\sigma(\mathsf{nonces}) = \sigma'(\mathsf{nonces})$. It follows that $A_j\sigma = A_j\sigma'$, that is $a_j = a'_j$ for all $1 \leq j \leq k$. We know that $N^0_{A_r}\sigma = \mathsf{n}^{a_r,0,s}$ and $N^0_{A_{r'}}\sigma' = \mathsf{n}^{a_{r'},0,s'}$. If $r = r'$ then we have in addition that $\mathsf{n}^{a_r,0,s'} = \mathsf{n}^{a_{r'},0,s'}$, thus $s = s'$ which is in contradiction with $\mathsf{sid} \neq \mathsf{sid}'$. Hence $r \neq r'$.

Finally, we prove the second point of Definition 9. Let i be the index of the analyzed transition $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ with $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$, m accepted, $m \neq \mathsf{fake}$ and $m = \widetilde{\mathsf{rcv}}^p_r\sigma$, where $f_i(\mathsf{sid}) = (\sigma, r, p+1)$.

Since tr is a valid trace, $H_{i-1} \vdash \widetilde{\mathsf{rcv}}^p_r\sigma$ holds. Consider a minimal proof associated with this deduction. We have $\widetilde{\mathsf{rcv}}^p_r = \{\!|\mathsf{rcv}^p_r, \llbracket \mathsf{rcv}^p_r, p'', \mathsf{nonces} \rrbracket_{\mathsf{sk}'(A_{r''})} |\!\}_{\mathsf{ek}'(A_r)}$ where $(r'', p'') = \mathcal{S}(r, p)$. Since $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ it follows that $A_{r''}\sigma$ is a non-corrupted agent. Hence $H_{i-1} \not\vdash \sigma(\mathsf{sk}'(A_{r''}))$. Thus the message $m_1 = \sigma(\llbracket \mathsf{rcv}^p_r, p'', \mathsf{nonces} \rrbracket_{\mathsf{sk}'(A_{r''})})$ was not obtained by a composition rule. Thus, in both cases: m obtained by a composition rule or by a decomposition rule, it follows that m_1 is a subterm of a term t' in H_{i-1}. The term t' was sent at some previous step. Thus there is $i' \leq i$ and $\mathsf{sid}' \in \mathsf{SId}_n$ such that $\alpha_{i'} = \mathbf{send}(\mathsf{sid}', m')$ for some $m' = \widetilde{\mathsf{rcv}}^{p'}_{r'}\sigma'$ and $t' = \widetilde{\mathsf{snt}}^{p'}_{r'}\sigma'$ where $f_{i'}(\mathsf{sid}') = (\sigma', r', p'+1)$. Suppose i' is the smallest such index, that is m_1 is not a subterm of a term of $H_{i'-1}$. We can then have two possibilities.

In the first one, m_1 is a subterm of $\mathsf{snt}^{p'}_{r'}\sigma'$. Since $\mathsf{snt}^{p'}_{r'}$ cannot contain the signature $\mathsf{sk}'(\cdot)$ (the source protocol is constructed over Σ), m_1 is a subterm of $X\sigma'$ where X is a variable of $\mathsf{snt}^{p'}_{r''}$. Hence m_1 is also a subterm of $\mathsf{rcv}^{p'}_{r'}\sigma'$ and moreover a subterm of $m' = \widetilde{\mathsf{rcv}}^{p'}_{r'}\sigma'$. But we have that $H_{i'-1} \vdash m'$. Consider m'_1 be the signed component of m'. Again m'_1 can be obtained only by a decomposition rule. Hence again by the locality lemma, m'_1 is a subterm of a term of $H_{i'-1}$. But m_1 is a subterm of m'_1. We have thus obtained a contradiction (i is not the smallest index such that m_1 is a subterm of a term of H) which means this case doesn't occur.

In the second possibility, $m_1 = \llbracket \mathsf{snt}^{p'}_{r'}, p', \mathsf{nonces} \rrbracket_{\mathsf{sk}'(A_{r'})}\sigma'$. It follows that $\mathsf{nonces}\,\sigma = \mathsf{nonces}\,\sigma'$ which implies that $\mathsf{sid} \sim \mathsf{sid}'$. We also have $p' = p''$. From $A_{r''}\sigma = A_{r'}\sigma'$ we obtain that $r' = r''$. Let r'_d, p'_d be such that $\mathcal{S}(r'_d, p'_d) = (r', p')$. They exists and are unique by the definition of executable protocols. But since $\mathcal{S}(r, p) = (r'', p'')$ and $(r'', p'') = (r', p')$ it follows that $r'_d = r$ and $p'_d = p$. Finally, since also $\mathsf{snt}^{p'}_{r'}\sigma' = \mathsf{rcv}^p_r\sigma$, we obtain that $m = \widetilde{\mathsf{snt}}^{p'}_{r'}$.

B.2 Proof of Lemma 2

Proof. First, note that all terms $t \in H$ are equal to $\widetilde{\mathsf{snt}}^p_r\sigma'$ for some f', sid', r and p with $f'(\mathsf{sid}') = (\sigma', r, p+1)$ and if $p \geq 1$ and $\mathsf{rcv}^p_r \neq \mathsf{init}$ then these terms are of the form $\{[m', [\![m', p, \sigma'(\mathsf{nonces})]\!]_{\mathsf{sk}'(a)}]\}_{\mathsf{ek}'(b)}$. Second, remark that it is sufficient to prove the desired property for all $t \in H$. The generalization to deducible messages follows easily. Hence it is sufficient to prove that whenever $X\sigma$ occurs in some $\widetilde{\mathsf{snt}}^p_r\sigma'$ then $\mathsf{rcv}^p_r \neq \mathsf{init}$, $p \geq 1$ and sid' is an honest session id.

Let $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ be a trace of $\widetilde{\Pi}$ and X be a variable of Π. We suppose without loss of generality that $X = N^j_{A_{\overline{r}}}$ for some $\overline{r} \in [k]$ and $j > 0$. Take (SId, f, H) an arbitrary global state of tr and let i be the index of this global state in tr. Consider an honest session id $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ and let $\mathsf{sid} = (s_0, r_0, \cdot)$ and $f(\mathsf{sid}) = (\sigma, r_0, \cdot)$.

We prove first that $N^j_{A_{\overline{r}}}\sigma$ is a nonce created in an honest session.

If $\overline{r} = r_0$ then we have that $N^j_{A_{\overline{r}}}\sigma = \mathsf{n}^{a_{\overline{r}},j,s}$. Suppose $\overline{r} \neq r_0$. This means that $N^j_{A_{\overline{r}}}$ was not initialized in sid by a **new** transition but by a $\alpha_{i_0} = \mathbf{send}(\mathsf{sid}, m)$ transition with $i_0 < i$, m accepted, $m \neq \mathsf{init}$ and $p_0 \geq 1$, where $f_{i_0}(\mathsf{sid}) = (\sigma_0, r_0, p_0 + 1)$. We have $N^j_{A_{\overline{r}}}\sigma = N^j_{A_{\overline{r}}}\sigma_0$. Let $(r_1, p_1) = \mathcal{S}(r_0, p_0)$. Since tr is an honest trace (by Lemma 1), there are $\mathsf{sid}_1 \in [\mathsf{sid}]$ and $i_1 < i_0$ such that $f_{i_1}(\mathsf{sid}_1) = (\sigma_1, r_1, p_1 + 1)$ for some substitution σ_1. We have $N^j_{A_{\overline{r}}}\sigma_0 = N^j_{A_{\overline{r}}}\sigma_1$. If $\overline{r} = r_1$ then $N^j_{A_{\overline{r}}}\sigma_1 = \mathsf{n}^{a_{\overline{r}},j,s_1}$ where $\mathsf{sid}_1 = (s_1, r_1, \cdot)$. Otherwise, continuing in the same way for at most i steps we will certainly find some index l, $0 \leq l < k$ such that $\overline{r} = r_l$ (this is because there are k different roles). Hence anyhow $N^j_{A_{\overline{r}}}\sigma_{\mathsf{sid}} = \mathsf{n}^{a_{\overline{r}},j,s_l}$. To ease the notation we denote it by n.

If $H_i = H_{i-1}$ then it is sufficient to prove the property for $i-1$. Hence consider that i is such that $H_i \setminus H_{i-1} \neq \emptyset$. It follows that $\alpha_i = \mathbf{send}(\mathsf{sid}', m)$ for some $\mathsf{sid}' \in \mathsf{SId}_i$ (clearly $\alpha_i \neq \mathbf{corrupt}$ since $H_1 \not\vdash \mathsf{n}$). Also $m = \widetilde{\mathsf{rcv}}^p_r\sigma'$ where $f_i(\mathsf{sid}') = (\sigma', r, p+1)$.

We reason by induction on i.

Suppose that i is the *smallest* index such that n occurs in a term of H_i. It follows that $\mathsf{n} \in St(\widetilde{\mathsf{snt}}^p_r\sigma')$. Then $\mathsf{n} \in St(Y\sigma')$ where Y is a variable of $\widetilde{\mathsf{snt}}^p_r$.

If Y is not a variable of $\widetilde{\mathsf{rcv}}^p_r$ then from the definition of executable protocols we know that $Y\sigma'$ is a new nonce or key, or an agent identity. Hence $Y = N^{j'}_{A_r}$ or $K^{j'}_{A_r}$ or A' for some j' and A'. That is, $Y\sigma'$ is a constant just like n; thus $Y\sigma' = \mathsf{n}$. Since $\mathsf{n} = \mathsf{n}^{a_r,j,s_l}$ it follows that $A_r\sigma' = a_{\overline{r}}$, $j = j'$ and $\mathsf{sid}' = (s_l, r, \cdot)$. Hence $\mathsf{sid}' = \mathsf{sid}_l$. This means that sid' is an honest session id. Suppose $p < 1$. Then $Y = N^0_{A_r}$ and thus $j = 0$ which is a contradiction. Hence $p \geq 1$ and thus in this case the property is true.

Otherwise, if Y is a variable of $\widetilde{\mathsf{rcv}}^p_r$ then $\mathsf{n} \in St(\widetilde{\mathsf{rcv}}^p_r\sigma')$, that is $\mathsf{n} \in St(m)$. Since $H_{i-1} \vdash m$, it follows that $\mathsf{n} \in St(H_{i-1})$, which is in contradiction with i being the smallest index such that $\mathsf{n} \in St(H_i)$.

Suppose now that i is *arbitrary*. We have $\mathsf{n} \in St(H_i \cup \{\widetilde{\mathsf{snt}}^p_r\sigma'\})$. For the occurrence of n in H_{i-1} then the conclusion follows by induction hypothesis. Consider an occurrence of n in $\widetilde{\mathsf{snt}}^p_r\sigma'$. If n occurs in $Y\sigma'$ where Y is not a variable of $\widetilde{\mathsf{rcv}}^p_r$ then, as in the previous paragraph, the conclusion simply follows. Suppose that n occurs in $Y\sigma$ where Y is a variable

of $\widetilde{\mathsf{rcv}}^{p}_{r}$. Then n occurs in m. Since m is deducible from H_{i-1} and in H_{i-1} all occurrences of n are as required by the induction hypothesis, it follows that the same thing happens in m. That is, $m = m''[\widetilde{\mathsf{snt}}^{p'}_{r'}\sigma'']$ and n occurs in $\widetilde{\mathsf{snt}}^{p'}_{r'}\sigma''$ where $f_i(\mathsf{sid}'') = (\sigma'', r', p'+1)$ for some honest session $\mathsf{sid}'' \in \mathsf{SId}_i$ and $p' \geq 1$. If the occurrence of $\widetilde{\mathsf{snt}}^{p'}_{r'}\sigma''$ in $m = \widetilde{\mathsf{rcv}}^{p}_{r}\sigma'$ is in $Y\sigma'$ then the conclusion follows, as this means that $\widetilde{\mathsf{snt}}^{p'}_{r'}\sigma''$ occurs in $\widetilde{\mathsf{snt}}^{p}_{r}\sigma'$. Otherwise it must be the case that $m = \widetilde{\mathsf{snt}}^{p'}_{r'}\sigma''$. Then $\sigma'(\mathsf{nonces}) = \sigma''(\mathsf{nonces})$. And since sid'' is an honest session id and $p' \geq 1$ we obtain that also sid' is also an honest session id and $p \geq 1$.

C Proof of Theorem 1

Consider an arbitrary security property $\phi \in \mathcal{L}'$ such that $\Pi \models^{p,1} \phi$.

Let $\mathsf{tr} \in \mathsf{Exec}(\widetilde{\Pi}) = (\mathsf{SId}_\iota, f_\iota, H_\iota)_{1\leq\iota\leq n}$. From Lemma 1 we know that $\mathsf{tr} \in \mathsf{Exec}^{\mathsf{h}}(\widetilde{\Pi})$. Also let $(\sigma, r, p) \in \mathcal{LS}_{r,p}(\mathsf{tr})$ such that $\mathsf{NC}(\mathsf{tr}, \sigma(A_l))$ holds for all $1 \leq l \leq k$. Hence there are an index ι with $1 \leq \iota \leq n$ and a session id $\mathsf{sid} \in \mathsf{SId}(\mathsf{tr})$ such that $f_\iota(\mathsf{sid}) = (\sigma, r, p)$. Moreover, $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$. We can suppose that $\iota \in \mathsf{Ix}(\mathsf{tr}, \mathsf{sid})$ because otherwise it would be easy to find another index which has this property.

Applying Lemma 3 we obtain that there are $\mathsf{tr}_0 \in \mathsf{Exec}^{p,1}(\Pi)$, $\mathsf{sid}_0 \in \mathsf{SId}(\mathsf{tr})$ and bijections $\mathcal{I}\colon \mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}]) \to \mathsf{Ix}(\mathsf{tr}_0, [\mathsf{sid}_0])$, $g\colon \mathsf{Ag}(\mathsf{sid}) \to \mathsf{Ag}(\mathsf{sid}_0)$ and $\varphi\colon [\mathsf{sid}] \to [\mathsf{sid}_0]$ satisfying certain properties. In particular, if we let $\mathsf{sid}_1 = \varphi(\mathsf{sid})$ and $\iota_0 = \mathcal{I}(\iota)$ then we have $f^0_{\iota_0}(\mathsf{sid}_1) = (\sigma_0, r, p)$ with $\sigma = \sigma_0 \circ g$.

Also, since tr_0 is an honest, single session trace by its definition, we have that sid_1 is an honest session id. Then $\mathsf{NC}(\mathsf{tr}_0, \sigma_0(A_l))$ is true for all $1 \leq l \leq k$. From the hypothesis we know that $[\![\phi(\mathsf{tr}_0)]\!] = 1$. Hence, since the left hand side of the implication holds it follows that also the right hand side holds for tr_0 and $(\sigma_0, r, p) \in \mathcal{LS}_{r,p}(\mathsf{tr}_0)$, that is for ι_0 and sid_1.

Fix an arbitrary i. What we have to prove depends on the form of the subformula in the right hand side of the implication.

Consider first that $\mathcal{Q}_i = \exists$. Since $[\![\phi(\mathsf{tr}_0)]\!] = 1$, there exist ι'_0 in $\mathsf{Ix}(\mathsf{tr}_0, [\mathsf{sid}_1])$ and $\mathsf{sid}'_0 \in \mathsf{SId}(\mathsf{tr}_0)$ such that the formulas $\tau^i_j(u^i_j, v^i_j)[^{\sigma_0}/_{\varsigma}][^{\sigma'_0}/_{\varsigma'}]$ hold for all $j \in J_i$, where $f^0_{\iota'_0}(\mathsf{sid}'_0) = (\sigma'_0, r_i, p_i)$. Let $\iota' = \mathcal{I}^{-1}(\iota'_0)$ and $\mathsf{sid}' = \varphi^{-1}(\mathsf{sid}'_0)$. Again by Lemma 3 we have that $f_{\iota'}(\mathsf{sid}') = (\sigma', r_i, p_i)$ with $\sigma' = \sigma'_0 \circ g$. Since σ, σ' are equal with σ_0, σ'_0 respectively, modulo the same bijective renaming g of agent identities, then it follows easily that $\tau^i_j(u^i_j, v^i_j)[^{\sigma}/_{\varsigma}][^{\sigma'}/_{\varsigma_i}]$ are true for all $j \in J_i$. Hence the formula $\exists\mathcal{LS}_{r_i,p_i}(\mathsf{tr}).\varsigma_i \bigwedge_{j\in J_i} \tau^i_j(u^i_j, v^i_j)$ is true.

Consider now that $\mathcal{Q}_i = \exists!$. The existence of ι' and sid' is assured as in the previous paragraph. Let $\varsigma(X)$ with X a nonce (or key) variable be a subterm in u^i_j or v^i_j for some $j \in J_i$ with $\tau^i_j \in \{=\}$.

Concerning uniqueness, assume there exist $\mathsf{sid}'' \in \mathsf{SId}(\mathsf{tr})$ and $\iota'' \in \mathsf{Ix}(\mathsf{tr})$ such that, in particular $(u^i_j = v^i_j)[^{\sigma}/_{\varsigma}][^{\sigma''}/_{\varsigma_i}]$, where $f_{\iota''} = (\sigma'', r_i, p_i)$. Consider an occurrence of $\varsigma(X)$ say in u^i_j, at position q. There is an occurrence q' in v^i_j with $q' \leq q$ such that $(v^i_j)|_{q'} = \varsigma(Y)$ or

$(v_j^i)|_{q'} = \varsigma_i(Y)$ where Y is a variable. Since we have uniqueness in the passive, single session case then $(v_j^i)|_{q'} = \varsigma_i(Y)$. Hence $X\sigma$ occurs in both $Y\sigma'$ and $Y\sigma''$.

If Y was received in session sid'' then there is an action $\alpha_{\iota_1''} = \mathbf{send}(\mathsf{sid}'', m)$ such that $m = \widetilde{\mathsf{rcv}}_{r_i}^{p'}\theta'$, $f_{\iota_1''}(\mathsf{sid}'') = (\theta', r_i, p'+1)$, $f_{\iota_1''-1}(\mathsf{sid}'') = (\theta, r_i, p')$ and θ was not defined on Y. We also have σ'' extends θ hence in particular $Y\sigma'' = Y\theta$. If Y was created (i.e. was initialized by a **new** action) in sid'' then it was also sent within some message $m = \widetilde{\mathsf{snt}}_{r_i}^{p'}\theta$, again with σ'' extending θ. In both cases, since m is deducible from the intruder's knowledge and $X\sigma$ occurs in m we can apply now Lemma 2 to obtain that sid'' is an honest session id and $\sigma''(\mathsf{nonces}) = \sigma(\mathsf{nonces})$. If Y was also received in session sid' then we can prove similarly that $\sigma'(\mathsf{nonces}) = \sigma(\mathsf{nonces})$. Intuitively, different role sessions can't be played by the same role (i.e. r_i) in the same protocol session hence $\mathsf{sid}' = \mathsf{sid}''$. Formally, this is obtained from the equality $N^0_{A_{r_i}}\sigma' = N^0_{A_{r_i}}\sigma''$ taking into account that $N^0_{A_{r_i}}$ was initialized in both sessions.

Finally consider that $\mathcal{Q}_i = \forall$.

Suppose that there are ι' and sid' such that $\tau_j^i(u_j^i, v_j^i)[^{\sigma}/_{\varsigma}][^{\sigma'}/_{\varsigma_i}]$ does not hold for some $j \in J_i$ where $f_{\iota'}(\mathsf{sid}') = (\sigma', r_i, p_i)$. That is $(u_j^i = v_j^i)[^{\sigma}/_{\varsigma}][^{\sigma'}/_{\varsigma_i}]$. Let $\varsigma(X)$ with X a nonce (or key) variable be a subterm in u_j^i (the case v_j^i is symmetric). Then, as before $X\sigma$ occurs in $Y\sigma''$ where $\sigma'' = \sigma$ or $\sigma'' = \sigma'$. If $\sigma'' = \sigma'$ then again using Lemma 2 it follows that sid' is an honest session and $\sigma'(\mathsf{nonces}) = \sigma(\mathsf{nonces})$. Hence $\mathsf{sid}' \in [\mathsf{sid}]$. Let $\iota_0' = \mathcal{I}(\iota')$ and $\mathsf{sid}_0' = \varphi(\mathsf{sid}_0')$. We have (from Lemma 3 again) that $\sigma' = \sigma_0' \circ g$. Hence $(u_j^i = v_j^i)[^{\sigma}/_{\varsigma}][^{\sigma'}/_{\varsigma_i}]$ which is a contradiction with the hypothesis for tr_0, ι_0 and σ_0. Hence the supposition made is false.

D Authentication properties

In this section we show how to use the logic defined in Section 4 to formally capture several authentication definitions proposed by Lowe [13]. For each property we give the informal definition and the corresponding formalisation in our logic.

In all the formulas below, we let p be the length of A's role.

D.1 Aliveness

Lowe's definition of aliveness:

> "We say that a protocol guarantees to an initiator A *aliveness* of another agent B if, whenever A (acting as an initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol."

And the corresponding formula:

$$\phi_1(\mathsf{tr}) \triangleq \forall \mathcal{LS}_{1,p}(\mathsf{tr}).\varsigma\ (\mathsf{NC}(\mathsf{tr}, A\varsigma) \wedge \mathsf{NC}(\mathsf{tr}, B\varsigma) \Rightarrow \exists \mathcal{LS}_{r,1}(\mathsf{tr}).\varsigma'\ (B\varsigma = B\varsigma'))$$

D.2 Weak agreement

Lowe's definition of weak agreement:

> "We say that a protocol guarantees to an initiator A *weak agreement* of another agent B if, whenever A (acting as an initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A."

And the corresponding formula:

$$\phi_2(\mathsf{tr}) \triangleq \forall \mathcal{LS}_{1,p}(\mathsf{tr}).\varsigma \ \big(\mathsf{NC}(\mathsf{tr}, A\varsigma) \wedge \mathsf{NC}(\mathsf{tr}, B\varsigma) \Rightarrow \exists \mathcal{LS}_{r,1}(\mathsf{tr}).\varsigma' \ (B\varsigma = B\varsigma') \wedge (A\varsigma = A\varsigma')\big)$$

D.3 Non-injective agreement

Lowe's definition of non-injective agreement:

> "We say that a protocol guarantees to an initiator A *non-injective agreement* of another agent B on a set of data items ds (where ds is a set of a set of free variables appearing in the protocol description) if, whenever A (acting as an initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables of ds."

And the corresponding formula:

$$\phi_3(\mathsf{tr}) \triangleq \forall \mathcal{LS}_{1,p}(\mathsf{tr}).\varsigma \ (\mathsf{NC}(\mathsf{tr}, A\varsigma) \wedge \mathsf{NC}(\mathsf{tr}, B\varsigma) \Rightarrow \exists \mathcal{LS}_{2,1}(\mathsf{tr}).\varsigma' \ (B\varsigma = B\varsigma') \wedge (A\varsigma = A\varsigma') \wedge \bigwedge_{1 \leq i \leq n} (X_i\varsigma = X_i\varsigma'))$$

where $ds = \{X_1, \ldots, X_n\}$.

D.4 (Injective) Agreement

Lowe's definition of (injective) agreement:

> "We say that a protocol guarantees to an initiator A *agreement* of another agent B on a set of data items ds if, whenever A (acting as an initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables of ds and each such run of A corresponds to a *unique* run of B."

And the corresponding formula:

$$\phi_4(\mathsf{tr}) \triangleq \forall \mathcal{LS}_{1,p}(\mathsf{tr}).\varsigma\ (\mathsf{NC}(\mathsf{tr}, A\varsigma) \wedge \mathsf{NC}(\mathsf{tr}, B\varsigma) \Rightarrow \exists! \mathcal{LS}_{2,1}(\mathsf{tr}).\varsigma'\ (B\varsigma = B\varsigma') \wedge (A\varsigma = A\varsigma') \wedge \bigwedge_{1 \leq i \leq n} (X_i\varsigma = X_i\varsigma'))$$

where $ds = \{X_1, \ldots, X_n\}$.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399