Myrto Arapinis, Stéphanie Delaune,
Steve Kremer

From one Session to many: Dynamic Tags for Security Protocols

Research Report LSV-08-17

May 2008

Laboratoire Spécification et Vérification

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

From one Session to many: Dynamic Tags for Security Protocols*

Myrto Arapinis, Stéphanie Delaune, and Steve Kremer

LSV, ENS Cachan & CNRS & INRIA, France

Abstract. The design and verification of cryptographic protocols is a notoriously difficult task, even in abstract Dolev-Yao models. This is mainly due to several sources of unboundedness (size of messages, number of sessions, ...). In this paper, we present a transformation which maps a protocol that is secure for a single session to a protocol that is secure for an unbounded number of sessions. The transformation is surprisingly simple, computationally light and works for arbitrary protocols that rely on usual cryptographic primitives, such as symmetric and asymmetric encryption as well as digital signatures.
Our result provides an effective strategy to design secure protocols: (i) design a protocol intended to be secure for one session (this can be verified with existing automated tools); (ii) apply our transformation and obtain a protocol which is secure for an unbounded number of sessions. A side-effect of this result is that we characterize a class of protocols for which secrecy for an unbounded number of sessions is decidable.

1 Introduction

Security protocols are small distributed programs which aim at guaranteeing properties such as confidentiality of data, authentication of participants, etc. The security of these protocols relies on the one hand on the security of cryptographic primitives, e.g., encryption and digital signatures, and on the other hand on the concurrency related aspects of the protocols themselves. History has shown that even if cryptography is supposed to be perfect, such as in the classical Dolev-Yao model [15], the correct design of security protocols is notoriously error-prone. See for instance [6] for an early survey on attacks. These difficulties come mainly from two sources of unboundedness: a protocol may be executed several times (we get several protocol *sessions*) and the attacker is allowed to build messages of unbounded size. Indeed, secrecy is known to be undecidable when an unbounded number of sessions is allowed (e.g. [8]), even if the message size is bounded [16]. However, when the number of sessions is bounded, and even without assuming a bounded message size, the problem becomes co-NP-complete [24]. Moreover, special purpose tools (e.g. [2]) exist which are highly efficient when the number of sessions is small.

* Work partly supported by ARA SSIA Formacrypt and CNRS/JST ICT "Cryptography and logic: Computer-checked security proofs"

In this paper we propose a protocol transformation which maps a protocol that is secure for a single session to a protocol that is secure for an unbounded number of sessions. This provides an effective strategy to design secure protocols: (i) design a protocol intended to be secure for one session (this can be efficiently verified with existing automated tools); (ii) apply our transformation and obtain a protocol which is secure for an unbounded number of sessions.

Our transformation. Our transformation can be informally described as follows. Suppose that Π is a protocol between k participants $A_1, \ldots, A_k$. The transformed protocol adds to Π a preamble

$$A_i \rightarrow All: \ A_i, N_i$$

in which each participant sends a freshly generated nonce N_i together with his identity to all other participants. This preamble allows each participant to compute a dynamic, session dependent *tag*: $\langle A_1, N_1 \rangle, \ldots, \langle A_k, N_k \rangle$. It will be used to tag each encryption and signature in Π. The transformation is surprisingly simple since it does not require any cryptographic protection of the preamble: the attacker is allowed to tamper with the messages exchanged during this phase and each participant may compute a different tag. Intuitively, the security relies on the fact that the participant A_i decides on a given tag for a given session which is ensured to be fresh as it contains his own freshly generated nonce N_i. In addition to its simplicity, the transformation is computationally light as it does not add any additional cryptographic application; it may merely increase the size of messages to be encrypted or signed. The transformation applies to a large class of protocols, which may use symmetric and asymmetric encryption, digital signature and hash function.

We may note that, *en passant*, we identify a class of tagged protocols for which security is decidable for an unbounded number of sessions. This directly follows from our main result as it stipulates that verifying security for a single session is sufficient to conclude security for an unbounded number of sessions.

Related Work. The kind of compiler we propose here has also been investigated in the area of cryptographic design in computational models, especially for the design of group key exchange protocols. For example, Katz and Yung [18] proposed a compiler which transforms a key exchange protocol secure against a passive eavesdropper into an authenticated protocol which is secure against an active attacker. Earlier work includes compilers for 2-party protocols (e.g. [4, 20]). In the symbolic model, recent works [12, 3] allow one to transform a protocol which is secure in a weak sense (roughly no attacker [12] or just a passive one [3] and a single session) into a protocol secure in the presence of an active attacker and for an unbounded number of sessions. All of these works share however a common drawback: the proposed transformations make heavy use of cryptography. This is mainly due to the fact the security assumptions made on the input protocol are rather weak. As already mentioned in [12], it is important, from an efficiency perspective to lighten the use of cryptographic primitives. In this work,

we succeed in doing so at the price of requiring stronger security guarantees on the input protocol. However, we argue that this is acceptable since efficient automatic tools exist to decide this security criterion on the input protocols.

We can also compare our work with existing decidable protocol classes for an unbounded number of sessions. An early result is the PTIME complexity result by Dolev *et al.* [14] for a restricted class, called *ping-pong* protocols. Other classes have been proposed by Ramanujam and Suresh [22, 23], and Lowe [19]. However, in both cases, temporary secrets, composed keys and ciphertext forwarding are not allowed which discards protocols, such as the Yahalom protocol (see also Section 4.3).

Outline of the paper. In Section 2 we describe the term algebra which is used to model protocol messages as well as the intruder capabilities to manipulate such terms. Then, in Section 3, we define the protocol language we use to model protocols. In Section 4 we formally describe our transformation and state our main transference result. Finally, in Section 5, we prove our main result before concluding. For readability, the proofs are given in the appendix.

2 Messages and intruder capabilities

2.1 Syntax

We use an abstract term algebra to model the messages of a protocol. For this we fix several disjoint sets. We consider an infinite set of *agents* $\mathcal{A} = \{\epsilon, a, b \ldots\}$ with the special agent ϵ standing for the attacker and an infinite set of *agent variables* $\mathcal{X} = \{x_A, x_B, \ldots\}$. We need also to consider an infinite set of *names* $\mathcal{N} = \{n, m \ldots\}$ and an infinite set of *variables* $\mathcal{Y} = \{y, z, \ldots\}$. We consider the following *signature* $\mathcal{F} = \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \langle\,\rangle, \mathsf{h}, \mathsf{pub}, \mathsf{priv}, \mathsf{shk}\}$. We suppose that the function symbols in $\mathcal{F}$ come with an arity: we have $\mathsf{ar}(f) = 2$ for $f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \langle\,\rangle, \mathsf{shk}\}$ and $\mathsf{ar}(f) = 1$ for the remaining symbols. These function symbols model cryptographic primitives: $\langle\,\rangle$ represents pairing, enc, enca and sign represent symmetric encryption, asymmetric encryption and signature respectively, h represents the application of a hash function whereas $\mathsf{pub}(a)$ and $\mathsf{priv}(a)$ are used to model public and private keys of an agent a, and $\mathsf{shk}(a, b)$ $(= \mathsf{shk}(b, a))$ is used to model the long-term symmetric key shared by agents a and b. Names are used to model atomic data such as nonces. The set of Terms is defined inductively by the following grammar:

$t, u, v, \ldots ::=$		term
	y	variable y
$\mid$	n	name n
$\mid$	x	agent variable x
$\mid$	a	agent a
$\mid$	$f(u)$	$f \in \{\mathsf{pub}, \mathsf{priv}\}$ and u is an agent or an agent variable
$\mid$	$\mathsf{shk}(u_1, u_2)$	u_1 and u_2 are agents or agent variables
$\mid$	$f(t_1, t_2)$	application of a symbol $f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \langle\,\rangle\}$
$\mid$	$\mathsf{h}(t)$	application of the hash symbol h

We sometimes write $\langle u_1, \ldots, u_n \rangle$ instead of writing $\langle u_1, \langle, \ldots, \langle u_{n-1}, u_n \rangle \ldots \rangle\rangle$. We say that a term is *ground* if it has no variable. We consider the usual notations for manipulating terms. We write $vars(t)$ (resp. $fresh(t)$, $agent(t)$) for the set of variables (resp. names, agents) occurring in t. We write $St(t)$ for the set of *subterms* of a term t and we define the set of *long-term secret keys* of a term t as

$$lgKeys(t) = \{\mathsf{priv}(u) \mid \mathsf{pub}(u) \text{ or } \mathsf{priv}(u) \in St(t)\} \cup \{\mathsf{shk}(u_1, u_2) \in St(t)\}.$$

Moreover, we define the set of *encrypted subterms* of a term t as

$$EncSt(t) = \{f(t_1, \ldots, t_n) \in St(t) \mid f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \mathsf{h}\}\}$$

and the set of *plaintexts* of a term t, denoted $plaintext(t)$, as the set of atomic data that occur in plaintext, i.e

- $plaintext(\mathsf{h}(u_1)) = plaintext(f(u_1, u_2)) = plaintext(u_1)$ for $f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}\}$
- $plaintext(\langle u_1, u_2 \rangle) = plaintext(u_1) \cup plaintext(u_2)$, and
- $plaintext(u) = \{u\}$ otherwise.

All these notions are extended to sets of terms and to other kinds of term containers as expected. We denote by $|S|$ the cardinality of a set S.

Example 1. Let $t = \mathsf{enc}(\langle n, \mathsf{pub}(a) \rangle, \mathsf{shk}(a, b))$. We have that $vars(t) = \emptyset$, i.e. t is ground, $fresh(t) = \{n\}$, $agent(t) = \{a, b\}$, $lgKeys(t) = \{\mathsf{priv}(a), \mathsf{shk}(a, b)\}$ and $plaintext(t) = \{n, \mathsf{pub}(a)\}$.

Substitutions are written $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where its *domain* is $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$. The substitution σ is *ground* if all the t_i are ground. The application of a substitution σ to a term t is written $\sigma(t)$ or $t\sigma$.

2.2 Intruder capabilities

We model the intruder's abilities to construct new messages by the deduction system given in Figure 1. The rules of the first line describe the *composition* rules and we refer to them as pairing, symmetric and asymmetric encryption, signature and hashing respectively. The second and the third lines describe the *decomposition* rules and the axiom. We refer to these rules as first and second projection, symmetric and asymmetric decryption respectively. The intuitive meaning of these rules is that an intruder can compose new messages by pairing, encrypting, signing and hashing previously known messages provided he has the corresponding keys. Conversely, he can decompose messages by projecting or decrypting provided he has the decryption keys. The intruder is also able to *verify* whether a digital signature $\mathsf{sign}(m, \mathsf{priv}(a))$ matches the message m provided he has the verification key $\mathsf{pub}(a)$. This does however not generate any new message and hence we do not model this capability in the deduction system. Our optional rule expresses that an intruder can retrieve the whole message from its signature. Whether this property holds depends on the actual signature scheme.

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{enc}(u, v)} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{enca}(u, v)} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{sign}(u, v)} \qquad \frac{T \vdash u}{T \vdash \mathsf{h}(u)}$$

$$\frac{T \vdash \langle u, v \rangle}{T \vdash u} \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash v} \qquad \frac{T \vdash \mathsf{enc}(u, v) \quad T \vdash v}{T \vdash u}$$

$$\frac{T \vdash \mathsf{enca}(u, \mathsf{pub}(v)) \quad T \vdash \mathsf{priv}(v)}{T \vdash u} \qquad \frac{T \vdash \mathsf{sign}(u, \mathsf{priv}(v))}{T \vdash u} \textit{ (optional)} \qquad \frac{}{T \vdash u}\, u \in T$$

$$\frac{}{T \vdash a}\, a \in \mathcal{A} \qquad \frac{}{T \vdash \mathsf{pub}(a)}\, a \in \mathcal{A} \qquad \frac{}{T \vdash \mathsf{shk}(a, \epsilon)}\, a \in \mathcal{A} \qquad \frac{}{T \vdash \mathsf{priv}(\epsilon)}$$

Fig. 1. Intruder deduction system.

Therefore we consider this rule to be optional. Our results are independent of whether this rule is part of the deduction system or not. The two first rules of the forth line model the fact that the intruder knows all public datas, *i.e.* agents and public keys. And finally, the two last rules of the last line model the fact that the intruder knows his own long-term secret keys.

Definition 1 (Deducible). *We say that a term u is* deducible *from a set of terms T, denoted $T \vdash u$, if there exists a tree such that the root is labeled by $T \vdash u$, every intermediate node is an instance of a rule given in Figure 1, and the leaves are of the form $T \vdash v$ with either $v \in \mathcal{A} \cup \{\mathsf{pub}(a), \mathsf{shk}(a, \epsilon) \mid a \in \mathcal{A}\} \cup \{\mathsf{priv}(\epsilon)\}$ or $v \in T$.*

Example 2. The term $\langle n, \mathsf{shk}(a, b) \rangle$ is deducible from $\{\mathsf{enc}(n, \mathsf{shk}(a, b)), \mathsf{shk}(a, b)\}$.

3 Models for security protocols

In this section, we give a language for specifying protocols and define their execution in presence of an active attacker. The model we consider is rather standard (see for instance [24, 11]).

3.1 Syntax

Our protocol model allows parties to exchange messages built from identities and randomly generated nonces using public key and symmetric encryption, digital signature and hashing. The individual behavior of each protocol participant is defined by a parametrized *role*. A role describes a sequence of *events*, i.e. a sequence of receiving and sending.

Definition 2 (Event, role and protocol). *An* event e *is either a* receive event*, denoted* $\mathsf{rcv}(u)$*, or a* send event*, denoted* $\mathsf{snd}(u)$*, where u is a term. A* role *is of the form* $\lambda x_1 \ldots \lambda x_k . \nu y_1 \ldots \nu y_p .\, \mathsf{seq}$*, where*

– $X = \{x_1, \ldots, x_k\}$ *is a set of agent variables, the parameters of the role, corresponding to the* k *participants of the protocol,*
– $Y = \{y_1, \ldots, y_p\}$ *is a set of variables representing the nonces generated by the role,*
– $\mathsf{seq} = \mathsf{e}_1; \mathsf{e}_2; \ldots; \mathsf{e}_\ell$ *is a sequence of events such that* $(vars(\mathsf{seq}) \setminus \{X\}) \subseteq \mathcal{Y}$*, i.e. all agents variables are parameters.*

Moreover, the sequence of events is such that for all i, $1 \leq i \leq \ell$*, we have that*

$$\mathsf{e}_i = \mathsf{snd}(u) \text{ implies } vars(u) \subseteq X \cup Y \cup \{vars(v) \mid \mathsf{e}_j = \mathsf{rcv}(v) \text{ and } j < i\}.$$

The set of roles is denoted by Roles*. The* length *of a role is the number of elements in its sequence of events. A* k-party protocol *is a mapping* $\Pi : [k] \to \mathsf{Roles}$*, where* $[k] = \{1, 2, \ldots, k\}$*.*

The last condition on variables in send events ensures that each variable which appears in a sent term is either one of the parameters, nonces, or is a variable which has been bound by a previous receive event.

Example 3. We illustrate our protocol syntax on the familiar Needham-Schroeder public-key protocol [21] that is informally described below:

$$\begin{array}{l} A \to B : \{\langle N_A, A\rangle\}_{\mathsf{pub}(B)} \\ B \to A : \{\langle N_A, N_B\rangle\}_{\mathsf{pub}(A)} \\ A \to B : \{N_B\}_{\mathsf{pub}(B)} \end{array}$$

Agent A sends to B his identity together with a freshly generated nonce, encrypted with the public key of B. Agent B replies by copying A's nonce and adds a fresh nonce N_B, encrypted by A's public key. The agent A acknowledges by forwarding B's nonce encrypted by B's public key.
In our syntax this protocol is modeled as follows.

$$\begin{array}{ll} \Pi(1) = \lambda x_A.\lambda x_B.\nu y. & \Pi(2) = \lambda x_A.\lambda x_B.\nu y'. \\ \quad \mathsf{snd}(\mathsf{enca}(\langle y, x_A\rangle, \mathsf{pub}(x_B))); & \quad \mathsf{rcv}(\mathsf{enca}(\langle z', x_A\rangle, \mathsf{pub}(x_B))); \\ \quad \mathsf{rcv}(\mathsf{enca}(\langle y, z\rangle, \mathsf{pub}(x_A))); & \quad \mathsf{snd}(\mathsf{enca}(\langle z', y'\rangle, \mathsf{pub}(x_A))); \\ \quad \mathsf{snd}(\mathsf{enca}(z, \mathsf{pub}(x_B))) & \quad \mathsf{rcv}(\mathsf{enca}(y', \mathsf{pub}(x_B))) \end{array}$$

Clearly, not all protocols written using the syntax above are meaningful. In particular, some of them might not be *executable*, e.g. the role $\mathsf{rcv}(\mathsf{h}(x)); \mathsf{snd}(x)$. Actually, our result holds also for non executable protocols. We will see in Section 4 that we only need to ensure a weaker hypothesis (see Theorem 1, Condition 2).

3.2 Scenarios and sessions

In our model, a session corresponds to the instantiation of one role. This means in particular that one "normal execution" of a k-party protocol requires k sessions, one per role. We may want to consider several sessions corresponding to different instantiations of a same role. Since the adversary may block, redirect and send new messages, all the sessions might be interleaved in many ways. Such an interleaving is captured by the notion of a *scenario*.

Definition 3 (Scenario). *A* scenario for a protocol $\Pi : [k] \to \mathsf{Roles}$ *is a sequence* $\mathsf{sc} = (r_1, sid_1) \cdots (r_n, sid_n)$ *where* r_i *is a role and* sid_i *a session identifier such that* $1 \leq r_i \leq k$, $sid_i \in \mathbb{N} \setminus \{0\}$, *the number of identical occurrences of a pair* (r, sid) *is smaller than the length of the role* r, *and* $sid_i = sid_j$ *implies* $r_i = r_j$.

The condition on identical occurrences ensures that a role cannot execute more events than it contains. The last condition ensures that a session number is not reused on other roles. We say that $(r, s) \in \mathsf{sc}$ if (r, s) is an element of the sequence sc.

Given a scenario and an instantiation for the parameters, we can define a *symbolic trace*, that is a sequence of events that corresponds to the interleaving of the scenario, for which the parameters have been instantiated, fresh nonces are generated and variables are renamed to avoid name collisions between different sessions.

Definition 4 (Symbolic trace). *Let* Π *be a* k*-party protocol with*

$$\Pi(j) = \lambda x_1^j. \ldots \lambda x_k^j. \nu y_1^j. \ldots \nu y_{p_j}^j. \mathsf{e}_1^j; \ldots; \mathsf{e}_{\ell_j}^j \qquad \text{for } 1 \leq j \leq k.$$

Given a scenario $\mathsf{sc} = (r_1, sid_1) \cdots (r_n, sid_n)$ *and a function* $\alpha : \mathbb{N} \to \mathcal{A}^k$, *the* symbolic trace $\mathsf{tr} = \mathsf{e}_1; \ldots; \mathsf{e}_n$ associated to sc and α *is defined as follows. Let* $q_i = |\{(r_j, sid_j) \in \mathsf{sc} \mid j \leq i, sid_j = sid_i\}|$, *i.e. the number of previous occurrences in* sc *of the session* sid_i. *We have* $q_i \leq \ell_{r_i}$ *and* $\mathsf{e}_i = (\mathsf{e}_{q_i}^{r_i})\sigma_{r_i, sid_i}$, *where*

- $\mathrm{dom}(\sigma_{r,sid}) = \{x_1^r, \ldots, x_k^r\} \cup \{y_1^r, \ldots, y_{p_r}^r\} \cup vars(\Pi(r))$,
- $\sigma_{r,sid}(y) = n_{y,sid}$ *if* $y \in \{y_1^r, \ldots, y_{p_r}^r\}$, *where* $n_{y,sid}$ *is a fresh name;*
- $\sigma_{r,sid}(x_i^r) = a_i$ *when* $\alpha(sid) = (a_1, \ldots, a_k)$;
- $\sigma_{r,sid}(z) = z_{sid}$ *otherwise, where* z_{sid} *is a fresh variable.*

A session sid *is said to be* honest *w.r.t.* α *when* $\alpha(sid) \subseteq (\mathcal{A} \setminus \{\epsilon\})^k$.

Intuitively, a session *sid* is honest if all of its participants, from the point of view of the agent playing the session *sid*, are honest (*i.e.* $\neq \epsilon$).

We define an operator K which associates to a symbolic trace tr the knowledge gained by the adversary, i.e. the set of (possibly non ground) terms that are sent in this symbolic trace. More precisely, we have that $\mathsf{K}(\mathsf{e}_1; \cdots; \mathsf{e}_\ell) = \bigcup_{1 \leq i \leq \ell} \mathsf{K}(\mathsf{e}_i)$ where $\mathsf{K}(\mathsf{rcv}(u)) = \emptyset$ and $\mathsf{K}(\mathsf{snd}(u)) = \{u\}$. This operator is useful in the following when we associate a constraint system to a symbolic trace.

3.3 Constraint systems

Constraint systems have been successfully used for verifying secrecy properties of finite scenarios (see for instance [24, 9, 13]). We now recall the definition of constraint systems. In the next section we discuss how secrecy for an *unbounded number of sessions* can be specified using (infinite) families of constraint systems.

Definition 5 (Constraint system). *A* constraint system C *is either* $\bot$ *or a finite sequence of expressions* $(T_i \Vdash u_i)_{1\leq i\leq n}$*, called* constraints*, where each* T_i *is a finite set of terms, called the* left-hand side *of the constraint, and each* u_i *is a term, called the* right-hand side *of the constraint, such that:*

- $T_i \subseteq T_{i+1}$ *for every* i *such that* $1 \leq i < n$*;*
- *if* $x \in vars(T_i)$ *for some* $1 \leq i \leq n$ *then* $\exists j < i$ *such that* $x \in vars(u_j)$*.*

A solution *of* C *is a ground substitution* θ *with* $\mathrm{dom}(\theta) = vars(C)$ *such that for every* $(T \Vdash u) \in C$*, we have that* $T\theta \vdash u\theta$*. The empty constraint system is always satisfiable whereas* $\bot$ *denotes an unsatisfiable system.*

The left-hand side of a constraint system usually represents the messages sent on the network. The second condition in Definition 5 says that each variable occurs first in some right-hand side. We denote by $\mathsf{maxlhs}(C)$ the maximal (for the inclusion) left-hand side of C, $\mathsf{minlhs}(C)$ its minimal (for the inclusion) left-hand side and by $\mathsf{rhs}(C)$ the set of its right-hand sides.

3.4 Secrecy

We now define the secrecy preservation problem for an unbounded number of sessions. Intuitively, a term m is secret if for all possible instantiations and scenarios, the ground term m' obtained when all parameters and nonces have been instantiated during an *honest* session remains secret. This definition leads us to consider an infinite family of constraint systems.

Definition 6 (Secrecy). *Let* Π *be a* k*-party protocol with*

$$\Pi(j) = \lambda x_1^j \dots \lambda x_k^j . \nu y_1^j \dots \nu y_{p_j}^j . \mathsf{e}_1^j \dots \mathsf{e}_{\ell_j}^j \qquad \text{for } 1 \leq j \leq k.$$

and let $m \in St(\mathsf{e}_i^r)$ *for some role* $1 \leq r \leq k$ *and* $1 \leq i \leq \ell_r$*. We say that* Π *preserves the secrecy of* m *for the initial knowledge* T_0*, if for any scenario* sc*, for any function* $\alpha : \mathbb{N} \to \mathcal{A}^k$ *and for any honest session* sid_h *(*i.e. $\alpha(sid_h) \in (\mathcal{A} \setminus \{\epsilon\})^k$*) such that* $(r, sid_h) \in \mathsf{sc}$*, the following constraint system is not satisfiable*

$$\{T_0 \cup \mathsf{K}(\mathsf{tr}_i) \Vdash u \mid \mathsf{tr}_i = \mathsf{tr}_{i-1}; \mathsf{rcv}(u) \text{ and } 0 \leq i \leq \ell\} \cup \{T_0 \cup \mathsf{K}(\mathsf{tr}) \Vdash m\sigma_{r,sid_h}\}$$

where tr *is the symbolic trace of length* ℓ *associated to* (sc, α)*,* tr_i *its prefix of length* i *and* σ_{r,sid_h} *is the substitution defined in Definition 4.*

Example 4. Consider again the Needham-Schroeder protocol. Let $\Pi(1)$ and $\Pi(2)$ be the two roles introduced in Example 3. This protocol is well-known to be insecure for any initial knowledge of the intruder T_0 w.r.t. $m = y'$. Let sid_1 and sid_2 be two session numbers such that $sid_1 \neq sid_2$ and consider the scenario $\mathsf{sc} = (1, sid_1)\,(2, sid_2)\,(1, sid_1)\,(2, sid_2)$ and the function α such that $\alpha(sid_1) = (a, \epsilon)$ and $\alpha(sid_2) = (a, b)$. The constraint system C associated to T_0, sc, α and

$m\sigma_{2,sid_2} = n_{y',sid_2}$ (according to Definition 6) is given below.

$$C := \left\{ \begin{array}{r} T_1 \stackrel{\mathsf{def}}{=} T_0,\ \mathsf{enca}(\langle n_{y,sid_1}, a\rangle, \mathsf{pub}(\epsilon)) \Vdash \mathsf{enca}(\langle z'_{sid_2}, a\rangle, \mathsf{pub}(b)) \\ T_2 \stackrel{\mathsf{def}}{=} T_1,\ \mathsf{enca}(\langle z'_{sid_2}, n_{y',sid_2}\rangle, \mathsf{pub}(a)) \Vdash \mathsf{enca}(\langle n_{y,sid_1}, z_{sid_1}\rangle, \mathsf{pub}(a)) \\ T_2,\ \mathsf{enca}(z_{sid_1}, \mathsf{pub}(\epsilon)) \Vdash \mathsf{enca}(n_{y',sid_2}, \mathsf{pub}(b)) \\ T_2,\ \mathsf{enca}(z_{sid_1}, \mathsf{pub}(\epsilon)) \Vdash n_{y',sid_2} \end{array} \right.$$

The substitution $\sigma = \{z'_{sid_2} \mapsto n_{y,sid_1}, z_{sid_1} \mapsto n_{y',sid_2}\}$ is a solution of C. However this protocol preserves the secrecy of m when considering one honest session for each role. This has been formally verified with the AVISPA tool [2]. Our transference result (described in the next section) will ensure that the protcol $\tilde{\Pi}$ (protocol obtained from Π by applying our transformation) is secure for an unbounded number of sessions.

4 Transformation of protocols

In Section 4.1 we define our transformation before we state our main result (Theorem 1) whose proof is postponed to Section 5. Finally, we discuss the tags which are used in our transformation in Section 4.3.

4.1 Our transformation

Given an input protocol Π, our transformation will compute a new protocol $\tilde{\Pi}$ which consists of two phases. During the first phase, the protocol participants try to agree on some common, dynamically generated, session identifier τ. For this, each participant sends a freshly generated nonce N_i together with his identity A_i to all other participants. (Note that if broadcast is not practical or if not all identities are known to each participant, the message can be send to some of the participants who forward the message.) At the end of this preamble, each participant computes a session identifier: $\tau = \langle\langle A_1, N_1\rangle, \ldots, \langle A_k, N_k\rangle\rangle$. Note that an active attacker may interfere with this initialization phase and may intercept and replace some of the nonces. Hence, the protocol participants do not necessarily agree on the same session identifier τ after this preamble. In fact, each participant computes his own session identifier, say τ_j. During the second phase, each participant j executes the original protocol in which the dynamically computed identifier is used for tagging each application of a cryptographic primitive. In this phase, when a participant opens an encryption, he will check that the tag is in accordance with the nonces he received during the initialization phase. In particular he can test the presence of his own nonce.

The transformation, using the informal Alice-Bob notation, is described below and relies on the tagging operation that is formally defined in Definition 7.

$$\Pi = \begin{cases} A_{i_1} \to A_{j_1} : m_1 \\ \vdots \\ A_{i_\ell} \to A_{j_\ell} : m_\ell \end{cases} \qquad \tilde{\Pi} = \begin{cases} \text{Phase 1} & \text{Phase 2} \\ A_1 \to All : \langle A_1, N_1 \rangle & A_{i_1} \to A_{j_1} : [m_1]_\tau \\ \vdots & \vdots \\ A_k \to All : \langle A_k, N_k \rangle & A_{i_\ell} \to A_{j_\ell} : [m_\ell]_\tau \\ \text{where } \tau = \langle \mathsf{tag}_1, \ldots, \mathsf{tag}_k \rangle \text{ with } \mathsf{tag}_i = \langle A_i, N_i \rangle \end{cases}$$

Note that, the Alice-Bob notation only represents what happens in a normal execution, i.e. with no intervention of the attacker. Of course, in such a situation, the participants agree on the same session identifier τ used in the second phase.

Definition 7 (Tagging). *Let u be two terms and tag be a k-tag for some integer k. The tagging of u with tag, denoted $[u]_{\mathsf{tag}}$, is inductively defined as follows:*

$$\begin{array}{lcll} [\langle u_1, u_2 \rangle]_{\mathsf{tag}} & = & \langle [u_1]_{\mathsf{tag}}, [u_2]_{\mathsf{tag}} \rangle & \\ [f(u_1, u_2)]_{\mathsf{tag}} & = & f(\langle \mathsf{tag}, [u_1]_{\mathsf{tag}} \rangle, [u_2]_{\mathsf{tag}}) & \text{for } f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}\} \\ [\mathsf{h}(u_1)]_{\mathsf{tag}} & = & \mathsf{h}(\langle \mathsf{tag}, [u_1]_{\mathsf{tag}} \rangle) & \\ [u]_{\mathsf{tag}} & = & u & \text{otherwise} \end{array}$$

This notion is extended to sequences of events as expected. We are now able to formally define our transformation.

Definition 8 (Protocol transformation). *Let Π be a k-party protocol such that*

$$\Pi(j) = \lambda x_1^j \ldots \lambda x_k^j . \nu y_1^j \ldots \nu y_{p_j}^j . \mathsf{seq}^j \qquad \text{for } 1 \leq j \leq k.$$

and the variables z_i^j $(1 \leq i, j \leq k)$ not appearing in Π (which can always be ensured by renaming variables in Π). The transformed protocol $\tilde{\Pi}$ is a k-party protocol defined as follows:

$$\tilde{\Pi}(j) = \lambda x_1^j \ldots \lambda x_k^j . \nu y_1^j \ldots \nu y_{p_j}^j . \nu z_j^j . \tilde{\Pi}^{\mathsf{init}}(j); [\mathsf{seq}^j]_{\tau_j} \qquad \text{for } 1 \leq j \leq k$$

where $\tau_j = \langle u_1^j, \ldots, u_k^j \rangle$ with $u_i^j = \langle x_i^j, z_i^j \rangle$, and

$$\tilde{\Pi}^{\mathsf{init}}(j) = \mathsf{rcv}(u_1^j) \cdot \ldots \cdot \mathsf{rcv}(u_{j-1}^j) \cdot \mathsf{snd}(u_j^j) \cdot \mathsf{rcv}(u_{j+1}^j) \cdot \ldots \cdot \mathsf{rcv}(u_k^j)$$

In the above definition, the protocol $\tilde{\Pi}^{\mathsf{init}}$ models the initialization phase and the variables z_i^j correspond to the nonces that are exchanged during this phase. In particular for the role j, the variable z_j^j is a freshly generated nonce while the other variables z_i^j, $i \neq j$, are expected to be bound to the other participant's nonces in the receive events. Remember also that the variables x_i^j are the role parameters which correspond to the agents. The tag computed by the j^{th} role in our transformation consists in the concatenation of the $k-1$ nonces received during the initialization phase together with the fresh nonce generated by the role j itself, i.e. z_j^j. We illustrate this transformation on the Needham-Schroeder protocol introduced in Section 2.

Example 5. Consider the Needham-Schroeder protocol described in Example 3. Applying our transformation we obtain a 2-party protocol $\tilde{\Pi}$. The role $\tilde{\Pi}(2)$ is described below. The role $\tilde{\Pi}(1)$ can be obtained in a similar way.

$$\begin{array}{rl}\tilde{\Pi}(2) = & \lambda x_A \lambda x_B.\nu y'.\nu z_B.\mathsf{rcv}(\langle x_A, z_A\rangle);\mathsf{snd}(\langle x_B, z_B\rangle);\\ & \mathsf{rcv}(\mathsf{enca}(\langle \tau, \langle z', x_A\rangle\rangle, \mathsf{pub}(x_B)));\\ & \mathsf{snd}(\mathsf{enca}(\langle \tau, \langle z', y'\rangle\rangle, \mathsf{pub}(x_A)));\\ & \mathsf{rcv}(\mathsf{enca}(\langle \tau, y'\rangle, \mathsf{pub}(x_B)))\end{array}$$

where $\tau = \langle\langle x_A, z_A\rangle, \langle x_B, z_B\rangle\rangle$. Note that Lowe's famous man-in-the-middle attack does not exist anymore on $\tilde{\Pi}$.

4.2 Main theorem

Theorem 1. *Let Π be a k-party protocol and $\tilde{\Pi}$ be the corresponding transformed protocol according to Definition 8. Let T_0 be a finite set of terms (the intruder's initial knowledge). Let $\mathsf{CK} = lgKeys(\Pi) \setminus T_0$ be the set of critical keys. Let $m \in St(\Pi(j))$ for some $1 \leq j \leq k$ and $\tilde{m}$ be its counterpart in $\tilde{\Pi}(j)$. Moreover, we assume that:*

1. *critical keys do not appear in plaintext, i.e. $\mathsf{CK} \cap plaintext(\Pi) = \emptyset$;*
2. *for any role $\lambda x_1 \ldots \lambda x_k.\nu y_1 \ldots \nu y_p.\mathsf{e}_1;\ldots;\mathsf{e}_\ell$, for any i such that e_i is a send event, for any variable $x \in plaintext(\mathsf{e}_i)$, we have $x \in \{x_1, \ldots, x_k, y_1, \ldots, y_p\}$ or $x \in plaintext(\mathsf{e}_j)$ for some receive event e_j such that $j \leq i$.*

If Π preserves the secrecy of m when considering one honest session of each role, then $\tilde{\Pi}$ preserves the secrecy of $\tilde{m}$ (for an unbounded number of honest and dishonest sessions).

We have already seen that our dynamic tagging is useful to avoid interaction between different sessions of the same role in an execution (see Example 5). A more detailed discussion follows in Section 4.3. Now, we discuss our other hypotheses.

Condition 1. The first condition requires that critical keys do not appear in plaintext. Consider the following protocol.

$$\begin{array}{rl}\Pi(1) = & \lambda x_A.\lambda x_B.\nu y.\\ & \mathsf{snd}(\mathsf{enca}(\mathsf{shk}(x_A, x_B), \mathsf{pub}(x_B)));\\ & \mathsf{snd}(\mathsf{enca}(\mathsf{priv}(x_A), \mathsf{pub}(x_B)));\\ & \mathsf{snd}(\mathsf{enc}(y, \langle \mathsf{shk}(x_A, x_B), \mathsf{priv}(x_A)\rangle))\end{array} \qquad \begin{array}{rl}\Pi(2) = & \lambda x_A.\lambda x_B.\\ & \mathsf{rcv}(\mathsf{enca}(z, \mathsf{pub}(x_B)));\\ & \mathsf{snd}(z)\end{array}$$

There is an attack on the secrecy of y (a freshly generated nonce by $\Pi(1)$) which requires two instances of $\Pi(2)$ played by b (with a). Indeed, after one session of the role $\Pi(2)$, the attacker may learn the long-term key $\mathsf{shk}(a, b)$ and after two sessions, he additionally learns the long-term key $\mathsf{priv}(a)$. Then the attacker is able to deduce $\langle \mathsf{shk}(a, b), \mathsf{priv}(a)\rangle$ and to decrypt the ciphertext containing the

secret. Such an attack can still be mounted after applying our transformation. However, we have to consider two instances of $\Pi(1)$ since two instances of $\Pi(2)$ will never accept to decrypt two ciphertexts issued from the same instance of $\Pi(1)$. This example is not a counter-example of our main result since condition 1 is not satisfied. Intuitively, Condition 1 disallows the sending of long-term secrets in a plaintext position (even under an encryption). This is generally satisfied by protocols and considered as a prudent engineering practice.

Condition 2. While this condition is necessary for our result to hold it is not really restrictive and only discards protocols that are intuitively non-executable. Indeed, it is easy to construct a protocol where critical keys are only used at key positions, but appear at a plaintext position with a sequence of the form $\mathsf{rcv}(\mathsf{enc}(x, z)); \mathsf{snd}(z)$. Hence, we can easily construct a counter-example like the one described above which satisfies Condition 1 but which does not satisfy Condition 2.

Our result states that if a protocol admits an attack then there exists an attack which only requires one honest session of each role. The situation is however slightly more complicated than it may seem at first sight since there is an infinite number of honest sessions, which one would need to verify separately. The following result by Comon and Cortier [7] however allows us to avoid this combinatorial explosion: when verifying secrecy properties it is sufficient to consider one single honest agent (which is allowed to "talk to herself"). Hence we can instantiate all the parameters with the same agent $a \in \mathcal{A} \setminus \{\epsilon\}$.

4.3 Other ways of tagging

First we notice that it is important for our tags to be *collaborative*, i.e., all participants do contribute by adding a fresh nonce. Our main theorem would for instance not hold if the tag consists of a single nonce chosen by only one of the participants.

Example 6. Consider the following 2-party protocol.

$$\begin{array}{l} A \to B : \mathsf{enca}(\langle A, K, N_a\rangle, \mathsf{pub}(B)),\ \ \mathsf{sign}(\mathsf{enca}(\langle A, N_a\rangle, \mathsf{pub}(B)), \mathsf{priv}(A)) \\ B \to A : N_a,\ \ \mathsf{enc}(N_b, K) \end{array}$$

We are interested in the secrecy of the nonce N_b generated by the role B. This protocol admits an attack (see below) that requires two honest sessions of the role B and works as follows. After one normal execution of the protocol, the attacker sends to B a message of the expected form. He forges the first part of the message using a fake key K_i whereas he reuses, for the second part, the signature eavesdropped during the execution of the first session. Then, B sends a fresh nonce N'_b encrypted under K_i. The attacker is now able to deduce the secret N'_b.

$$\begin{array}{rl} & A \to B : \mathsf{enca}(\langle A, K, N_a\rangle, \mathsf{pub}(B)),\ \ \mathsf{sign}(\mathsf{enca}(\langle A, N_a\rangle, \mathsf{pub}(B)), \mathsf{priv}(A)) \\ & B \to A : N_a, \mathsf{enc}(N_b, K) \\ I(A) \to & B : \mathsf{enca}(\langle A, K_i, N_a\rangle, \mathsf{pub}(B)),\ \ \mathsf{sign}(\mathsf{enca}(\langle A, N_a\rangle, \mathsf{pub}(B)), \mathsf{priv}(A)) \\ & B \to A : N_a, \mathsf{enc}(N'_b, K_i) \end{array}$$

Note that such an attack (requiring two sessions of the role B) exists

- even if we add static tags, i.e. a different constant inside each encryption and signature, and
- even if the tag contains a fresh nonce generated by A.

This example illustrates that the contribution of B is crucial to prevent such attacks.

We have also considered an alternate, slightly different transformation. It is defined as the previous transformation (Definition 8), but does not include the identities in the tag, i.e., the tag is simply the sequence of nonces. In that case we obtain a different result: if a protocol admits an attack then there exists an attack which only requires one (not necessarily honest) session for each role. In this case, we need to additionally check for attacks that involve a session engaged with the intruder. On the example of the Needham-Schroeder protocol the man in the middle attack is not prevented by this weaker tagging scheme. However, the result requires one to also consider one dishonest session for each role, hence including the attack scenario.

Finally, different kinds of tags have also been considered in [1, 5, 22]. However these tags are *static* and have a different aim. While our dynamic tagging scheme avoids confusing messages from different sessions, these static tags avoid confusing different messages inside a same session. However, these tags do not prevent that a same message is reused in two different sessions. Under some additional assumptions (e.g. no temporary secret, no ciphertext forwarding), several decidability results [23, 19] have been obtained by showing that it is sufficient to consider one session per role. In the framework we consider here, the question whether such static tags would be sufficient to obtain decidability is still an open question (see [1]). Note that Example 6 relies on a temporary secret.

In a similar way, static tags have also been used by Heather et al. [17] to avoid type confusion attacks.

5 Proof of our main result

The proof of our main result is closely tied to a particular procedure for solving constraint systems (see [13, 10, 11]). We therefore first give a brief description of this procedure before outlining the proof itself.

5.1 Constraint solving procedure

The procedure we consider for constraint solving uses simplification rules that transform a given constraint system into another, simpler one. Such a simplification step is denoted $C \rightsquigarrow_\sigma C'$ where σ is a substitution that has been applied to C during this step. When the substitution is omitted it implicitly refers to the identity function. We also write $C \rightsquigarrow_\sigma^n C'$ for a sequence of n steps where σ

is the composition of the substitutions applied at each step. The procedure has been shown to be sound, complete and terminating [10]. This means that the procedure always terminates after a finite number of steps resulting either in $\bot$ when no solution exists or in a constraint system in *solved form*. A constraint system is in solved form when the right-hand side of each constraint is a variable. In that case the constraint system can be trivially solved by substituting each of these variables by a term t_0 such that $\mathsf{minlhs}(C) \vdash t_0$, e.g. ϵ. Moreover, if there is a simplification sequence $C \rightsquigarrow_\sigma^n C'$ where C' is in solved form then σ (extended to substitute the remaining variables by t_0) is a solution of C. The inference system we consider (see Figure 1) is slightly different from the one used in [10]. However, it is not difficult to show that the procedure described below is still sound, complete and terminating.

For the purpose of our proof, we decorate each term t by a pair (r, s) which denotes the role number and the session identifier in which t originated. The resulting term $t^{(r,s)}$ is called a labeled term. By convention, terms in T_0 (the initial knowledge of the intruder) are labeled with $(0, 0)$. These decorations do not influence the procedure but provide additional information that is useful in the proof. We could have added these decorations in Definition 4 when constructing the symbolic trace, but it would increase notational clutter and harm readability.

$$\mathsf{R}_1: \quad C \wedge T \Vdash u^{(r,sid)} \rightsquigarrow C \qquad \text{if } T \cup \{x \mid T' \Vdash x \in C, T' \subsetneq T\} \vdash u$$

$$\mathsf{R}_2: \quad C \wedge T \Vdash u^{(r,sid)} \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash (u\sigma)^{(r,sid)}$$
$$\text{if } \sigma = \mathrm{mgu}(t, u) \text{ where } t \in St(T),\ t \neq u,\ t, u \text{ are neither variables nor pairs}$$

$$\mathsf{R}_3: \quad C \wedge T \Vdash u^{(r,sid)} \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash (u\sigma)^{(r,sid)}$$
$$\text{if } \sigma = \mathrm{mgu}(t_1, t_2),\ t_1, t_2 \in St(T),\ t_1 \neq t_2,\ t_1, t_2 \text{ are neither variables nor pairs}$$

$$\mathsf{R}_4: \quad C \wedge T \Vdash u^{(r,sid)} \rightsquigarrow \bot \qquad \text{if } vars(T, u) = \emptyset \text{ and } T \not\vdash u$$

$$\mathsf{R}_5: \quad C \wedge T \Vdash f(u, v)^{(r,sid)} \rightsquigarrow C \wedge T \Vdash u^{(r,sid)} \wedge T \Vdash v^{(r,sid)} \qquad \text{for } f \in \{\langle\rangle, \mathsf{enc}, \mathsf{enca}, \mathsf{sign}\}$$

$$\mathsf{R}_6: \quad C \wedge T \Vdash \mathsf{h}(u)^{(r,sid)} \rightsquigarrow C \wedge T \Vdash u^{(r,sid)}$$

Fig. 2. Simplification rules

We extend all notations defined on terms to labeled terms, by providing a session identifier as an additional argument: e.g. $vars(T, sid) = \bigcup_{t^{(i,sid)} \in T} vars(t)$.

5.2 Proof of Theorem 1

Theorem 1 is proved by contradiction. Assume that $\tilde{\Pi}$ admits an attack. This means that there exists a scenario sc, a function $\alpha : \mathbb{N} \to \mathcal{A}^k$ and an honest session sid_h such that the associated constraint system C (according to Definition 6) is satisfiable. We will prove that the constraint system C' associated to sc' (the subsequence of sc where we only consider some particular sessions,

say S, chosen according to the tag τ_{sid_h} involved in the encrypted subterms of the honest session sid_h) and α, is also satisfiable. Intuitively $sid \in S$ if and only if the nonce generated during the initialisation phase of the session sid appears in tag τ_{sid_h} at the expected position, i.e. at the r^{th} position where r is the role associated to the session identifier sid. Initially, we have that $C' = C|_S$ according to the following definition.

Definition 9 (constraint system $C|_S$). *Let C be a constraint system and S a set of session identifiers, we define the restriction of C to S as follows*

$$C|_S := \{T|_S \Vdash u^{(r,s)} \mid s \in S \text{ and } (T \Vdash u^{(r,s)}) \in C\},$$

where $T|_S = \{v^{(r,s)} \in T \mid s \in S \cup \{0\}\}$.

We want to ensure that the simplification steps are stable by restriction to some well-chosen set S of sessions (see Lemma 2). While this property does not hold for general constraint systems we show below that it is the case for *well-formed* constraint systems.

Our notion of *well-formed* constraint systems relies on some additional definitions. A *k-tag* is a term composed of k pairs $\langle a, u\rangle$ where $a \in \mathcal{A}$ and $u \in \mathsf{Terms}$. We say that a term is *k-tagged* if all its encrypted subterms are tagged with a k-tag , i.e.

$$\forall u \in EncSt(t),\ \exists \mathsf{tag}, u_1, \ldots, u_n.\ u = f(\langle \mathsf{tag}, u_1\rangle, \ldots, u_n)$$

We denote by $tags(t)$ the set of k-tags which occur in a tagging position in t.

Definition 10 (Well-formed). *A constraint system $C = T_1 \Vdash u_1 \wedge \ldots \wedge T_n \Vdash u_n$ is* well-formed *w.r.t. a set T of k-tagged labeled terms if the following hold:*

1. $\mathsf{maxlhs}(C) \subseteq T$ *and* $\mathsf{rhs}(C) \subseteq St(T)$*;*
2. *the constraint system C satisfies the* plaintext origination property*, i.e. if $x \in vars(plaintext(T_i))$ then $\exists\, j < i$ such that $x \in vars(plaintext(u_j))$;*
3. *for all sid we have that $|tags(T, sid)| \leq 1$;*
4. *for all sid_1, sid_2, such that $tags(T, sid_1) \neq tags(T, sid_2)$, we have that*

$$vars(T, sid_1) \cap vars(T, sid_2) = \emptyset \ \wedge\ fresh(T, sid_1) \cap fresh(T\, sid_2) = \emptyset.$$

Intuitively, Condition 1 states that the terms in C are k-tagged. Condition 2 ensures that any variable appearing as a plaintext has been previously received in a plaintext position. Condition 3 says that all terms that originated in the same session have the same tag. Finally, Condition 4 ensures that sessions that are currently tagged in different ways in C use different variables and different nonces. Note that terms issued from different sessions are not necessarily tagged differently. First, we show that the simplification rules maintain well-formedness.

Lemma 1. *Let T be a set of k-tagged labeled terms, and C be a constraint system well-formed w.r.t. T. Let D be a constraint system, σ be a substitution and n be an integer such that $C \rightsquigarrow_\sigma^n D$. Then, we have that D is well-formed w.r.t. $T\sigma$ and, for any session sid, we have that $tags(T\sigma, sid) = (tags(T, sid))\sigma$.*

Relying on Lemma 1 we show that there exists a derivation from $C|_S$ to a constraint system in solved form, i.e. the existence of an attack involving only sessions in S.

Lemma 2. *Let* CK *be a set of long-term keys,* T *be a set of* k*-tagged labeled terms and* C *be a constraint system well-formed w.r.t.* T *and such that*

- $lgKeys(C) \setminus \mathsf{CK} \subseteq \mathsf{minlhs}(C)$ *and those terms are labeled with* $(0, 0)$*, and*
- $\mathsf{CK} \cap plaintext(\mathsf{maxlhs}(C)) = \emptyset$.

Let D *be a constraint system that is satisfiable,* σ *be a substitution and* n *be an integer such that* $C \rightsquigarrow^n_\sigma D$*. Let* tag *be a* k*-tag and* $Sid(\mathsf{tag}) = \{sid \mid tags(T\sigma, sid) = \mathsf{tag}\}$*. Then, there exists* $m \leq n$ *such that* $C|_{Sid(\mathsf{tag})} \rightsquigarrow^m_{\sigma|_Y} D|_{Sid(\mathsf{tag})}$ *where* $Y = \bigcup_{sid \in Sid(\mathsf{tag})} vars(T, sid)$.

This lemma is proved by induction. The proof is technical and the details can be found in the appendix. We consider the different rules and distinguish several cases depending on whether the terms involved are labeled with a session identifier in S or not. For instance, the rules R_5 (resp. R_4 and R_6) are mimiced by using the same instance of the same rule when the labeled term $u^{(r,sid)}$ (right hand side of the constraint) is such that $sid \in S$. Otherwise, we keep the constraint system unchanged. For the rules R_2 (resp. R_3) the key point is that terms which are tagged differently cannot be unified and do not share any variables nor fresh names (this is due to well-formedness). Thus, the unifier σ used in this step involved two terms labeled by sid_1 and sid_2 that are either both in S or both not in S. This is due to the fact that, after application of σ, these two terms will be tagged in the same way and thus by definition of S, have the same status. If both are in S, we can apply the same rule. If none of them is in S, we show that σ has no effect and we keep the constraint system unchanged. The case of the rule R_1 can also be proved in a similar way.

In order to pursue the proof of Theorem 1, we apply Lemma 2 on the derivation $C \rightsquigarrow^n_\sigma D$ witnessing the existence of an attack on $\tilde{\Pi}$ and we consider $S = \{sid \mid tags(T\sigma, sid) = tags(T\sigma, sid_h)\}$, i.e. the sessions that are tagged in the same way that sid_h. We obtain that $C|_S$ can also reach a constraint system in solved form, namely $D|_S$. Moreover, the satisfiability of $C|_S$ witnesses the fact that there is an attack on $\tilde{\Pi}$ that only involves sessions in S. Now, in order to conclude, it remains to show that:

1. S does not contain two distinct sessions that execute the same role. Intuitively, this comes from the fact that sessions in S are tagged in the same way (after application of σ) and this is not possible for two distinct sessions that execute the same role. Indeed, the fresh nonce generated by different sessions of the same role ensures that their tag are distinct.
2. S only contains honest sessions. First sid_h is an honest session by definition of the secrecy property. Second, since the names of the agents engaged in a role occur in the tag and sessions in S are tagged in the same way as the session sid_h, we conclude that this property is also true for any $sid \in S$.

Thus, there is an attack on $\tilde{\Pi}$ that involves at most one honest session of each role. To conclude, it is easy to see that this attack can also be mounted on the protocol Π.

6 Conclusion and future work

In this paper we proposed a protocol transformation which transfers security against active adversaries for one protocol session to security against active adversaries for an unbounded number of sessions. By strengthening the security hypothesis on the input protocols with respect to other existing work, our obtained transformation is surprisingly simple and computationally light. As security for a single session can efficiently be verified with existing tools, this result provides an appealing protocol design strategy.

Our current result applies to transfer secrecy properties. As future work we foresee to extend the scope of our result to other security properties, e.g. authentication or more challenging equivalence based properties. We also plan to extend the result to other intruder theories. We foresee that such results require new proof methods which are not based on the decision procedure as in this paper, but directly on the semantics. Another challenging topic for future research is to obtain more fine-grained characterizations of decidable classes of protocols for an unbounded number of sessions. The new insights gained by our work seem to be a good starting point to extract the conditions needed to reduce the security for an unbounded number of sessions to security for a finite number of sessions.

Acknowledgments. We would like to thank Yassine Lakhnech for discussions that initiated this work as well as Hubert Comon-Lundh, Véronique Cortier, Joshua Guttman and Ralf Küsters for their helpful comments.

References

1. M. Arapinis and M. Duflot. Bounding messages for free in security protocols. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 376–387. Springer, 2007.
2. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The Avispa tool for the automated validation of internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *Lecture Notes on Computer Science*. Springer, 2005.
3. D. Beauquier and F. Gauche. How to guarantee secrecy for cryptographic protocols. *CoRR*, abs/cs/0703140, 2007.
4. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *Proc. 30th Annual ACM Symposium on the Theory of Computing (STOC'98)*, pages 419–428. ACM Press, 1998.

5. B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Proc. Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *Lecture Notes on Computer Science*, pages 136–152. Springer, 2003.
6. J. Clark and J. Jacob. A survey of authentication protocol literature. http://www.cs.york.ac.uk/ jac/papers/drareviewps.ps, 1997.
7. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. *Science of Computer Programming*, 50(1-3):51–71, 2004.
8. H. Comon-Lundh and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. In *Theoretical Computer Science*, volume 331, pages 143–214, 2005.
9. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. 18th Annual Symposium on Logic in Computer Science (LICS'03)*, pages 271–280. IEEE Comp. Soc. Press, 2003.
10. V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'07)*, volume 4855 of *Lecture Notes on Computer Science*, pages 352–363. Springer, Dec. 2007.
11. V. Cortier and S. Delaune. Safely composing security protocols. Research Report LSV-08-06, Laboratoire Spécification et Vérification, ENS Cachan, France, Mar. 2008. 39 pages.
12. V. Cortier, B. Warinschi, and E. Zalinescu. Synthesizing secure protocols. In *Proc. 12th European Symposium On Research In Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes on Computer Science*, pages 406–421. Springer, 2007.
13. V. Cortier and E. Zalinescu. Deciding key cycles for security protocols. In *Proc. 13th Inter. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes on Computer Science*, pages 317–331. Springer, 2006.
14. D. Dolev, S. Even, and R. M. Karp. On the security of ping-pong protocols. In *Proc. Advances in Cryptology (CRYPTO'82)*, pages 177–186, 1982.
15. D. Dolev and A. C. Yao. On the security of public key protocols. In *Proc. of the 22nd Symposium on Foundations of Computer Science (FOCS'81)*, pages 350–357. IEEE Comp. Soc. Press, 1981.
16. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. Workshop on Formal Methods and Security Protocols*, 1999.
17. J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proc. 13th Computer Security Foundations Workshop (CSFW'01)*, pages 255–268. IEEE Comp. Soc. Press, 2000.
18. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proc. 23rd Annual International Cryptology Conference (CRYPTO'03)*, volume 2729 of *Lecture Notes on Computer Science*, pages 110–125. Springer, 2003.
19. G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(1), 1999.
20. A. J. Mayer and M. Yung. Secure protocol transformation via "expansion": From two-party to groups. In *Proc. 6th ACM Conference on Computer and Communications Security (CCS'99)*, pages 83–92. ACM Press, 1999.
21. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, 1978.

22. R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *Lecture Notes on Computer Science*, pages 363–374. Springer-Verlag, 2003.
23. R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.
24. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299(1-3):451–475, 2003.

A Proof of Lemma 1

Before we prove Lemma 1, we first introduce some useful lemmas.

Lemma 3. *Let T be a set of k-tagged labeled terms such that*

- $\forall sid,\ |tags(T, sid)| \leq 1$,
- $\forall sid, sid'. tags(T, sid) \neq tags(T, sid') \Rightarrow$

$$vars(T, sid) \cap vars(T, sid') = \emptyset \quad \wedge \quad fresh(T, sid) \cap fresh(T, sid') = \emptyset,$$

Let $u_1^{(r_1, sid_1)}, u_2^{(r_2, sid_2)} \in T$ and $t_1 \in EncSt(u_1),\ t_2 \in EncSt(u_2)$ unifiable, with $\sigma = \mathrm{mgu}(t_1, t_2)$. Then

- $tags(T\sigma, sid_1) = tags(T\sigma, sid_2) = \sigma(tags(T, sid_1)) = \sigma(tags(T, sid_2))$
- $\forall x.\ \sigma(x)$ *is k-tagged.*

Lemma 3 can be proved by induction on the algorithm that computes the most general unifier.

Moreover we can easily show that,

i. $|tags(T\sigma, sid_i)| = 1,\ i \in \{1, 2\}$
ii. $\forall x \in \mathrm{dom}(\sigma). tags(\sigma(x)) \subseteq tags(T\sigma, sid_1)\ (= tags(T\sigma, sid_2))$
iii. $\forall sid.\ tags(T\sigma, sid) \neq tags(T\sigma, sid_i) \Rightarrow tags(T, sid) \neq tags(T, sid_1) \wedge tags(T, sid) \neq tags(T, sid_2) \qquad i \in \{1, 2\}$

Lemma 4. *Let T be a set of k-tagged labeled terms such that*

1. $\forall sid$ *we have that* $|tags(T, sid)| \leq 1$, *and*
2. $\forall sid,\ sid'$, *such that* $tags(T, sid) \neq tags(T, sid')$, *we have that*

$$vars(T, sid) \cap vars(T, sid') = \emptyset \quad \wedge \quad fresh(T, sid) \cap fresh(T, sid') = \emptyset.$$

Let $u_1^{(r_1, sid_1)}, u_2^{(r_2, sid_2)} \in T$, and $t_1 \in EncSt(u_1)$, $t_2 \in EncSt(u_2)$ unifiable with $\sigma = \mathrm{mgu}(t_1, t_2)$. Then $T\sigma$ is a set of k-tagged labeled terms such that

3. $\forall sid$ *we have that* $|tags(T\sigma, sid)| \leq 1$, *and*
4. $\forall sid,\ sid'$ *such that* $tags(T\sigma, sid) \neq tags(T\sigma, sid')$, *we have that*

$$vars(T\sigma, sid) \cap vars(T\sigma, sid') = \emptyset \quad \wedge \quad fresh(T\sigma, sid) \cap fresh(T\sigma, sid') = \emptyset.$$

Proof. We first show that $T\sigma$ is a set of k-tagged labeled terms, *i.e.* that for all $t \in T\sigma,\ t$ is k-tagged, or equivalently that for all $u \in EncSt(T\sigma),\ u = f(\langle \mathsf{tag}, u_1 \rangle, \ldots, u_n)$ for some k-tag tag. We have that $u \in EncSt(T\sigma)$ is equivalent to:

- either, $\exists v \in EncSt(T)$ such that $u = v\sigma$
- or, $\exists x \in vars(T)$ such that $u \in EncSt(\sigma(x))$

We show that in both cases $u = f(\langle \mathsf{tag}, u_1 \rangle, \ldots, u_n)$ for some k-tag tag.

– Suppose there exists $v \in \mathit{EncSt}(T)$ such that $u = v\sigma$. Then v is of the form $f(\langle \mathsf{tag}', v_1\rangle, \ldots, v_n)$ with $f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \mathsf{h}\}$, tag' some k-tag, and $u = f(\langle \mathsf{tag}'\sigma, v_1\sigma\rangle, \ldots, v_n\sigma)$, *i.e.* $\mathsf{tag} = \mathsf{tag}'\sigma$. We show by induction on k that $\mathsf{tag}'\sigma$ is a k-tag. If $k = 1$, tag' is by definition of the form $\langle a, t\rangle$ with $a \in \mathcal{A}$, and $t \in \mathsf{Terms}$. So, $\mathsf{tag}'\sigma = \langle a, t\sigma\rangle$ and $t\sigma \in \mathsf{Terms}$. Thus, $\mathsf{tag} = \mathsf{tag}'\sigma$ is a 1-tag. Suppose now that $k > 1$, then tag' is, by definition of a k-tag, of the form $\langle\langle a, t\rangle, \mathsf{tag}''\rangle$, for some $a \in \mathcal{A}$, some $t \in \mathsf{Terms}$, and some $(k-1)$-tag tag''. So $\mathsf{tag}'\sigma = \langle\langle a, t\sigma\rangle, \mathsf{tag}''\sigma\rangle$ and by induction it is the case that $\mathsf{tag}''\sigma$ is a $(k-1)$-tag. So $\mathsf{tag}'\sigma = \langle\langle a, t\sigma\rangle, \mathsf{tag}''\sigma\rangle$ is by definition a k-tag.
– Suppose there exists $x \in \mathit{vars}(T)$ such that $u \in \mathit{EncSt}(\sigma(x))$. By lemma 3 we know that $\sigma(x)$ is k-tagged. By definition of a k-tagged term we have that $u = f(\langle \mathsf{tag}, u_1\rangle, \ldots, u_n)$ for some k-tag tag.

Now we proceed with the proof of points 3 and 4 of the lemma. We recall here that $\mathit{tags}(T\sigma, \mathit{sid}_1) = \mathit{tags}(T\sigma, \mathit{sid}_2)$ (Lemma 3).

3. let sid be a session. We need to distinguish two cases
   (a) $\mathit{tags}(T\sigma, \mathit{sid}) \neq \mathit{tags}(T\sigma, \mathit{sid}_1)$.
   Thus $\mathit{tags}(T, \mathit{sid}) \neq \mathit{tags}(T, \mathit{sid}_1)$, and $\mathit{tags}(T, \mathit{sid}) \neq \mathit{tags}(T, \mathit{sid}_2)$. By condition 2 we know that $\mathit{vars}(T, \mathit{sid}) \cap \mathit{vars}(T, \mathit{sid}_1) = \emptyset$, and that $\mathit{vars}(T, \mathit{sid}) \cap \mathit{vars}(T, \mathit{sid}_2) = \emptyset$, thus $\mathit{vars}(T, \mathit{sid}) \cap \mathrm{dom}(\sigma) = \emptyset$ from which it follows that $\mathit{tags}(T\sigma, \mathit{sid}) = \mathit{tags}(T, \mathit{sid})$, thus $|\mathit{tags}(T\sigma, \mathit{sid})| \leq 1$
   (b) $\mathit{tags}(T\sigma, \mathit{sid}) = \mathit{tags}(T\sigma, \mathit{sid}_1)$.
   By Lemma 3 we have that

$$\mathit{tags}(T\sigma, \mathit{sid}) = \sigma(\mathit{tags}(T, \mathit{sid}_1))$$

   and since $|\mathit{tags}(T, \mathit{sid}_1)| \leq 1$ from condition 1 of the present lemma, $|\mathit{tags}(T\sigma, \mathit{sid})| = |\sigma(\mathit{tags}(T, \mathit{sid}_1))| \leq 1$.

4. Let sid and sid' be two sessions such that $\mathit{tags}(T\sigma, \mathit{sid}) \neq \mathit{tags}(T\sigma, \mathit{sid}')$, and let r be the role played by session sid, and r' the role played by session sid'. We distinguish three cases:
   (a) $\mathit{tags}(T\sigma, \mathit{sid}) \neq \mathit{tags}(T\sigma, \mathit{sid}_1)$ and $\mathit{tags}(T\sigma, \mathit{sid}') \neq \mathit{tags}(T\sigma, \mathit{sid}_1)$.
   So it immediately follows that , $\mathit{tags}(T, \mathit{sid}) \neq \mathit{tags}(T, \mathit{sid}_1)$, $\mathit{tags}(T, \mathit{sid}) \neq \mathit{tags}(T, \mathit{sid}_2)$, $\mathit{tags}(T, \mathit{sid}') \neq \mathit{tags}(T, \mathit{sid}_1)$, and $\mathit{tags}(T, \mathit{sid}') \neq \mathit{tags}(T, \mathit{sid}_2)$, and thus

$$\mathit{vars}(T, \mathit{sid}) \cap \mathrm{dom}(\sigma) = \emptyset \qquad \mathit{vars}(T, \mathit{sid}') \cap \mathrm{dom}(\sigma) = \emptyset$$

   implying that:

   – $\mathit{vars}(T\sigma, \mathit{sid}) = \mathit{vars}(T, \mathit{sid})$ – $\mathit{vars}(T\sigma, \mathit{sid}') = \mathit{vars}(T, \mathit{sid}')$
   – $\mathit{fresh}(T\sigma, \mathit{sid}) = \mathit{fresh}(T, \mathit{sid})$ – $\mathit{fresh}(T\sigma, \mathit{sid}') = \mathit{fresh}(T, \mathit{sid}')$

   Now, since $\mathit{tags}(T\sigma, \mathit{sid}) \neq \mathit{tags}(T\sigma, \mathit{sid}')$, it is necessarily the case that $\mathit{tags}(T, \mathit{sid}) \neq \mathit{tags}(T, \mathit{sid}')$, so by hypothesis

$$\mathit{vars}(T, \mathit{sid}) \cap \mathit{vars}(T, \mathit{sid}') = \emptyset \quad \text{and} \quad \mathit{fresh}(T, \mathit{sid}) \cap \mathit{fresh}(T, \mathit{sid}') = \emptyset$$

So,

$$vars(T\sigma, sid) \cap vars(T\sigma, sid') = vars(T, sid) \cap vars(T, sid') = \emptyset$$

and,

$$fresh(T\sigma, sid) \cap fresh(T\sigma, sid') = fresh(T, sid) \cap fresh(T, sid') = \emptyset$$

(b) $tags(T\sigma, sid) = tags(T\sigma, sid_1)$. Thus,

$$vars(T\sigma, sid) \subseteq vars(T, sid) \cup vars(T, sid_1) \cup vars(T, sid_2)$$

and,

$$fresh(T\sigma, sid) \subseteq fresh(T, sid) \cup fresh(T, sid_1) \cup fresh(T, sid_2).$$

Now, since $tags(T\sigma, sid) \neq tags(T\sigma, sid')$ it is necessarily the case that $tags(T, sid) \neq tags(T, sid')$, and thus $vars(T, sid) \cap vars(T, sid') = \emptyset$ and $fresh(T, sid) \cap fresh(T, sid') = \emptyset$.
For the same reason, we also have that $vars(T, sid_1) \cap vars(T, sid') = \emptyset$, $vars(T, sid_2) \cap vars(T, sid') = \emptyset$, $fresh(T, sid_1) \cap fresh(T, sid') = \emptyset$, and $fresh(T, sid_2) \cap fresh(T, sid') = \emptyset$.
Moreover, we have that $vars(T, sid') \cap vars(T, sid_1) = \emptyset$ and also that $vars(T, sid') \cap vars(T, sid_2) = \emptyset$. This implies $vars(T, sid') \cap \mathrm{dom}(\sigma) = \emptyset$, and by mimicking the argument of (4a), we also have that $vars(T\sigma, sid') = vars(T, sid')$, $fresh(T\sigma, sid') = fresh(T, sid')$. Finally,

$$\begin{aligned} & vars(T\sigma, sid) \cap vars(T\sigma, sid') \\ = \ & vars(T\sigma, sid) \cap vars(T, sid') \\ \subseteq \ & (vars(T, sid) \cup vars(T, sid_1) \cup vars(T, sid_2)) \cap vars(T, sid') \\ = \ & \emptyset \text{ from above discussion} \end{aligned}$$

and

$$\begin{aligned} & fresh(T\sigma, sid) \cap fresh(T\sigma, sid') \\ = \ & fresh(T\sigma, sid) \cap fresh(T, sid') \\ \subseteq \ & (fresh(T, sid) \cup fresh(T, sid_1) \cup fresh(T, sid_2)) \cap fresh(T, sid') \\ = \ & \emptyset \text{ from above discussion} \end{aligned}$$

(c) $tags(T\sigma, sid') = tags(T\sigma, sid_1)$ - analogous to the previous case. □

Lemma 5 ([11], Lemma 11). *Let T be a set of terms and u be a term such that $T \vdash u$. Then, we have that $plaintext(u) \subseteq plaintext(T)$.*

Lemma 1. *Let T be a set of k-tagged labeled terms, and C be a constraint system well-formed w.r.t. T. Let D be a constraint system, σ be a substitution and n be an integer such that $C \rightsquigarrow_\sigma^n D$. Then, we have that D is well-formed w.r.t. $T\sigma$ and, for any session sid, we have that $tags(T\sigma, sid) = (tags(T, sid))\sigma$.*

Proof. We prove this result by induction on the length n of the derivation.
Base Case: $n = 0$. In such a case, we have that $D = C$ and we easily conclude.

Induction Step: $n \geq 1$. In such a case, there exists a constraint system E and substitutions σ_1, σ_2 such that

$$C \rightsquigarrow^{1}_{\sigma_1} E \rightsquigarrow^{n-1}_{\sigma_2} D \quad \text{and} \quad \sigma = \sigma_1\sigma_2$$

In order to conclude by induction, we show by case analysis on the simplification rule R involved in the first step of derivation that E is well-formed w.r.t. $T\sigma_1$, and that $\mathit{tags}(T\sigma_1, \mathit{sid}) = \sigma_1(\mathit{tags}(T, \mathit{sid}))$ for any session sid. Then, it will be easy to conclude by applying our induction hypothesis.

Case $\mathsf{R} = \mathsf{R}_1$. In such a case, there exists C', U, u, r and sid such that

$$C = (C' \wedge U \Vdash u^{(r,\mathit{sid})}) \rightsquigarrow_{\mathsf{id}} C' = E$$

and, $U \cup \{x \mid U' \Vdash x \in C,\ U' \subsetneq U\} \vdash u$. The only delicate point in order to conclude in C''s well-formedness with respect to $T\sigma_1 = T$, is to show that C' satisfies the plaintext origination property. Suppose it doesn't, then we are necessarily in the situation where $i/$ $\exists y \in \mathit{plaintext}(u)$, $ii/$ $\exists (V \Vdash v^{(r',\mathit{sid}')}) \in C$ *s.t.* $U \subsetneq V$ and $y \in \mathit{plaintext}(V)$, and $iii/$ $\forall (W \Vdash w^{(r'',\mathit{sid}'')}) \in C'$, *s.t.*$W \subsetneq V$ and $y \notin \mathit{plaintext}(w)$. But from lemma 5, we know that since $U \cup \{x \mid U' \Vdash x \in C,\ U' \subsetneq U\} \vdash u$, then $\mathit{plaintext}(u) \subseteq U \cup \{x \mid U' \Vdash x \in C,\ U' \subsetneq U\}$, and in particular $y \in U \cup \{x \mid U' \Vdash x \in C,\ U' \subsetneq U\}$:

- if $y \in U$, then since $(U \Vdash u) \in C$, and C satisfies the plaintext origination property, necessarily exists $(W \Vdash w) \in C$ such that $y \in \mathit{plaintext}(w)$ and $W \subsetneq U$. Thus, $(W \Vdash w) \in C'$ with $W \subsetneq V$ and $y \in \mathit{plaintext}(w)$ exist and contradicts $iii/$
- if $y \in \{x \mid U' \Vdash x \in C,\ U' \subsetneq U\} \vdash u$, then exists $(W \Vdash y) \in C$ with $W \subsetneq U$, and thus exists $(W \Vdash y) \in C'$ with $W \subsetneq V$. Here again we are confronted to a contradiction with $iii/$.

Thus, C' satisfies the plaintext origination property. Moreover, since we have that σ_1 is the identity and E is a subset of deducibility constraints of C, we easily conclude for the other conditions of well-formednes. Finally, since $\sigma_1 = \mathsf{id}$, $\forall \mathit{sid}$

$$\mathit{tags}(T\sigma_1, \mathit{sid}) = \mathit{tags}(T, \mathit{sid}) = \mathit{tags}(T, \mathit{sid})\sigma_1.$$

Case $\mathsf{R} = \mathsf{R}_4$. In such a case, we have that σ_1 is the identity and E is a subset of deducibility constraints of C. Thus, we easily conclude.

Case $\mathsf{R} = \mathsf{R}_5$ *or* R_6. In this case, it is also easy to conclude since σ_1 is the identity and we only decompose a term in $\mathsf{rhs}(C)$. Thus, we have that E is well-formed w.r.t. $T\sigma_1 = T$ and that $\mathit{tags}(T\sigma_1, \mathit{sid}) = \mathit{tags}(T, \mathit{sid}) = \mathit{tags}(T, \mathit{sid})\sigma_1$.

Case $\mathsf{R} = \mathsf{R}_2$ *or* R_3. We give the details below when $\mathsf{R} = \mathsf{R}_2$, the case $\mathsf{R} = \mathsf{R}_3$ can be done in a similar way. In such a case, there exist C', U, u, v, w, r, and sid such that:

$$C = (C' \wedge U \Vdash u^{(r,\mathit{sid})}) \rightsquigarrow_{\sigma_1} (C'\sigma_1 \wedge U\sigma_1 \Vdash u\sigma_1^{(r,\mathit{sid})}) = E$$

where $\sigma_1 = \mathrm{mgu}(u, v)$ and $v \in \mathit{EncSt}(w)$. It is easy to see that $\mathsf{maxlhs}(E) \subseteq T\sigma_1$ and $\mathsf{rhs}(E) \subseteq \mathit{St}(T\sigma_1)$. Then, we conclude for the remaining conditions of well-formedness thanks to Lemma 4. Moreover, the plaintext origination property is stable under unification, and thus $\mathit{tags}(T\sigma_1, \mathit{sid}) = \mathit{tags}(T, \mathit{sid}) = \mathit{tags}(T, \mathit{sid})\sigma_1$. □

B Proof of Lemma 2

Before proving Lemma 2, we recall the following lemma from [11] that allow us to ensure that critical keys never appear in plaintext.

Lemma 6 ([11], Lemma 12). *Let* CK *be a set of long-term keys,* C *be a constraint system satisfying the plaintext origination property (see Definition 10) such that* $\mathsf{CK} \cap \mathit{plaintext}(\mathsf{maxlhs}(C)) = \emptyset$ *and* σ *be a substitution. If* $C\sigma$ *is satisfiable, then*
$\mathsf{CK} \cap \mathit{plaintext}(\mathsf{maxlhs}(C\sigma)) = \emptyset$.

Lemma 2. *Let* CK *be a set of long-term keys,* T *be a set of* k*-tagged labeled terms and* C *be a constraint system well-formed w.r.t.* T *and such that*

- $\mathit{lgKeys}(C) \setminus \mathsf{CK} \subseteq \mathsf{minlhs}(C)$ *and those terms are labeled with* $(0, 0)$*, and*
- $\mathsf{CK} \cap \mathit{plaintext}(\mathsf{maxlhs}(C)) = \emptyset$.

Let D *be a constraint system that is satisfiable,* σ *be a substitution and* n *be an integer such that* $C \rightsquigarrow_\sigma^n D$*. Let* tag *be a* k*-tag and* $\mathit{Sid}(\mathsf{tag}) = \{\mathit{sid} \mid \mathit{tags}(T\sigma, \mathit{sid}) = \mathsf{tag}\}$*. Then, there exists* $m \leq n$ *such that* $C|_{\mathit{Sid}(\mathsf{tag})} \rightsquigarrow_{\sigma|_Y}^m D|_{\mathit{Sid}(\mathsf{tag})}$ *where* $Y = \bigcup_{\mathit{sid} \in \mathit{Sid}(\mathsf{tag})} \mathit{vars}(T, \mathit{sid})$.

Proof. Let tag be a k-tag and $S = \mathit{Sid}(\mathsf{tag})$ (as defined in Lemma 2). From now on, we assume that there exists $s \in S$. Otherwise the result is obvious. We show the result by induction on the length n of the derivation $C \rightsquigarrow_\sigma^n D$.

Base Case: $n = 0$. In such a case, we have $C = D$, and thus $C|_S = D|_S$. This allows us to conclude.

Induction Step: $n \geq 1$. In such a case, there exists a constraint system E and substitutions σ_1, σ_2 such that

$$C \rightsquigarrow_{\sigma_1}^1 E \rightsquigarrow_{\sigma_2}^{n-1} D \quad \text{and} \quad \sigma = \sigma_1\sigma_2$$

In order to conclude by induction, we show by case analysis on the simplification rule R involved in the first derivation step that we are in one of the following cases:

- either $C|_S = E|_S$,
- or $C|_S \rightsquigarrow_{\sigma_1|_S}^{\mathsf{R}} E|_S$.

and E is a constraint system well-formed w.r.t. $T\sigma_1$ (thanks to Lemma 1), $lgKeys(E) \setminus \mathsf{CK} \subseteq \mathsf{minlhs}(E)$ and those terms are labeled with $(0,0)$ (obvious); and lastly we have that $\mathsf{CK} \cap plaintext(\mathsf{maxlhs}(E)) = \emptyset$ (thanks to Lemma 6). This will allows us to conclude by applying our induction hypothesis.

Case $\mathsf{R} = \mathsf{R}_4$. It would lead to $D = \bot$ which contradicts the fact that the constraint system D is supposed to be satisfiable.

Case $\mathsf{R} = \mathsf{R}_5$ *or* R_6. We give the details below when $\mathsf{R} = \mathsf{R}_5$, the case $\mathsf{R} = \mathsf{R}_6$ can be done in a similar way. In such a case there exist C', U, u, v, f, r and sid such that

$$C \stackrel{\mathsf{def}}{=} C' \wedge U \Vdash f(u,v)^{(r,sid)} \rightsquigarrow_{\sigma_1} C' \wedge U \Vdash u^{(r,sid)} \wedge U \Vdash v^{(r,sid)} \stackrel{\mathsf{def}}{=} E.$$

- If $sid \notin S$, then $(U|_S \Vdash u^{(r,sid)}) \notin C|_S$, and thus $C|_S = E|_S$.
- If $sid \in S$, then we have that $C|_S \rightsquigarrow E|_S$:
  $C'|_S \wedge U|_S \Vdash f(u,v)^{(r,sid)} \rightsquigarrow C'|_S \wedge U|_S \Vdash u^{(r,sid)} \wedge U|_S \Vdash v^{(r,sid)} E|_S$. To conclude we notice that $\sigma_1|_S = \mathsf{id}|_S = \mathsf{id}$.

Case $\mathsf{R} = \mathsf{R}_2$ *or* R_3. We give the details below when $\mathsf{R} = \mathsf{R}_2$, the case $\mathsf{R} = \mathsf{R}_3$ can be done in a similar way. In such a case, there exist C', U, u, v, w, r, and sid such that:

$$C \stackrel{\mathsf{def}}{=} (C' \wedge U \Vdash u^{(r,sid)}) \rightsquigarrow_{\sigma_1} (C'\sigma_1 \wedge U\sigma_1 \Vdash u\sigma_1^{(r,sid)}) \stackrel{\mathsf{def}}{=} E$$

where $\sigma_1 = \mathrm{mgu}(u,v)$ and $v \in EncSt(w)$ with $w^{(r',sid')} \in T$ for some r' and sid'. We distinguish two cases:

- If $sid \notin S$, then $(U|_S \Vdash u^{(r,sid)}) \notin C|_S$, and thus $E|_S = C'\sigma_1|_S$. By hypothesis, we have that C is well-formed w.r.t. T. By definition of S and since $sid \notin S$, we have that $sid' \notin S$. We also have that $tags(T, sid) \neq tags(T, s)$ (resp. $tags(T, sid') \neq tags(T, s)$) for any $s \in S$. By Lemma 1, we infer that $vars(T, sid) \cap vars(T, s) = vars(T, sid') \cap vars(T, s) = \emptyset$ for any $s \in S$. Thus, $\mathrm{dom}(\sigma_1) \cap vars(T, s)$ for any $s \in S$. So

$$C'|_S = (C'|_S)\sigma_1 = E|_S \text{ and } \sigma_1|_S = \mathsf{id}$$

- If $sid \in S$, since $\sigma_1 = \mathrm{mgu}(u,v)$ then $tags(T\sigma_1, sid) = tags(T\sigma_1, sid')$. Thus $tags(T\sigma, sid') = \mathsf{tag}$ and from the definition of S, we have that $w^{(r',sid')} \in U|_S$. Hence, we have that
  - $C|_S = C'|_S \wedge U|_S \Vdash u^{(r,sid)}$, and
  - $E|_S = C'\sigma_1|_S \wedge U\sigma_1|_S \Vdash (u\sigma_1)^{(r,sid)}$.

  So, we easily conclude that $C|_S \rightsquigarrow_{\sigma_1} E|_S$, and by definition of $\sigma_1|_S$ that $\sigma_1|_S = \sigma_1$.

Case $\mathsf{R} = \mathsf{R}_1$. In such a case, there exist C', U, u, r and sid such that

$$C \stackrel{\mathsf{def}}{=} C' \wedge U \Vdash u^{(r,sid)} \rightsquigarrow C' \stackrel{\mathsf{def}}{=} E.$$

If $sid \notin S$, then we have that $C|_S = E|_S$. Otherwise, that is if $sid \in S$ we show that $U|_S \cup \{x \mid (U'|_S \Vdash x) \in C|_S,\ U'|_S \subsetneq U|_S\} \vdash u^{(r,sid)}$. This is done by induction (on the prooftree witnessing $U \cup \{x \mid (U' \Vdash x) \in C,\ U' \subsetneq U\} \vdash u^{(r,sid)}$). If u is public data of the form a or $\mathsf{pub}(a)$, for some $a \in \mathcal{A}$, and since $T \vdash u$ for all T (see last line of Figure 1), necessarily we have that

$$U|_S \cup \{x \mid (U'|_S \Vdash x) \in C|_S,\ U'|_S \subsetneq U|_S\} \vdash u^{(r,sid)}.$$

Otherwise, the only case for which we do not conclude by induction is when the last rule of the proof is the axiom, *i.e.* when $u^{(r',sid')} \in U$ for some r' and sid'. We then need to analyze the form of u:

- *Case: u is a variable.* We know from the fact that C is well-formed w.r.t. T that $tags(T, sid) = tags(T, sid')$ $(= \mathsf{tag})$ (condition 4 of the definition of well-formed), and thus from the definition of $C|_S$ we necessarily have $u^{(r',sid')} \in U|_S$. So, $U|_S \vdash u^{(r,sid)}$.
- *Case: $u \in fresh(T, sid)$.* Again, from the fact that C is well-formed w.r.t. T, we know that $tags(T, sid) = tags(T, sid')$ $(= \mathsf{tag})$ (condition 4 of the definition of well-formed), and from the definition of $C|_S$ we have $u^{(r',sid')} \in U|_S$. So, $U|_S \vdash u^{(r,sid)}$.
- *Case: $u \in \mathsf{CK}$.* By hypothesis we have that $\mathsf{CK} \cap plaintext(\mathsf{maxlhs}(C)) = \emptyset$, but $u \in U \subseteq \mathsf{maxlhs}(C)$ would imply that $u \in plaintext(\mathsf{maxlhs}(C))$ (according to Lemma 5).This contradicts the fact that $\mathsf{CK} \cap plaintext(\mathsf{maxlhs}(C)) = \emptyset$.
- *Case: $u \in lgKeys(C) \setminus \mathsf{CK}$.* By hypothesis, we have that $lgKeys(C) \setminus \mathsf{CK} \subseteq \mathsf{minlhs}(C)$ and those terms are labeled with $(0, 0)$. Thus, we have that $u \in U|_S$, thus $U|_S \vdash u^{(r,sid)}$.
- *Case: $u = f(u_1, u_2)$ with $f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}\}$.* In such a case u_1 is of the form $\langle \mathsf{tag}', u_1' \rangle$ for some k-tag tag' and u_1', thus $tags(T, sid') = \mathsf{tag}'$ and by definition of S, we have that $sid' \in S$; thus $u^{(r',sid')} \in U|_S$. So, $U|_S \vdash u^{(r,sid)}$.
- *Case: $u = \mathsf{h}(u')$.* This case is analogous to the previous one.
- *Case: $u = \langle u_1, u_2 \rangle$.* If there exists $u' \in EncSt(u)$ then we necessarily have $tags(u) = \mathsf{tag}'$ for some k-tag tag', and thus $u^{(r',sid')} \in U|_S$ allowing us to conclude as in the previous case. Otherwise, u is obtained only by applying the pairing function over some atomic data $\mathcal{D}$. In this case, we conclude by showing that each $d \in \mathcal{D}$ is deducible from $U|_S$. The reasoning is similar to the ones performed for the above first two cases together with the case where d is public data.

In both cases ($sid \in S$ or $sid \notin S$), since $\sigma_1 = \mathsf{id}$, obviously $\sigma|_S = \mathsf{id}$, we can thus conclude the proof. □

C Proof of Theorem 1

Theorem 1. *Let Π be a k-party protocol and $\tilde{\Pi}$ be the corresponding transformed protocol according to Definition 8. Let T_0 be a finite set of terms (the intruder's initial knowledge). Let $\mathsf{CK} = \mathit{lgKeys}(\Pi) \setminus T_0$ be the set of critical keys. Let $m \in \mathit{St}(\Pi(j))$ for some $1 \leq j \leq k$ and $\tilde{m}$ be its counterpart in $\tilde{\Pi}(j)$. Moreover, we assume that:*

1. *critical keys do not appear in plaintext, i.e. $\mathsf{CK} \cap \mathit{plaintext}(\Pi) = \emptyset$;*
2. *for any role $\lambda x_1 \ldots \lambda x_k.\nu y_1 \ldots \nu y_p.\mathsf{e}_1; \ldots; \mathsf{e}_\ell$, for any i such that e_i is a send event, for any variable $x \in \mathit{plaintext}(\mathsf{e}_i)$, we have $x \in \{x_1, \ldots, x_k, y_1, \ldots, y_p\}$ or $x \in \mathit{plaintext}(\mathsf{e}_j)$ for some receive event e_j such that $j \leq i$.*

If Π preserves the secrecy of m when considering one honest session of each role, then $\tilde{\Pi}$ preserves the secrecy of $\tilde{m}$ (for an unbounded number of honest and dishonest sessions).

Proof. Suppose $\tilde{\Pi}$ admits an attack on $\tilde{m} \in \mathit{St}(\tilde{\Pi}(r))$ for some role r and some initial knowledge T_0 . This means that there exist a scenario sc, a function $\alpha : \mathbb{N} \to \mathcal{A}^k$, an honest session sid_h (with $(r, \mathit{sid}_h) \in \mathsf{sc}$) such that the corresponding constraint system C (as given in Definition 6) with $T_0 \cup \mathsf{K}(\mathsf{tr}) \Vdash \tilde{m}\sigma_{r,\mathit{sid}_h}$ (where tr is the symbolic trace associated to sc and α) is satisfiable. We first show that there is an attack on $\tilde{\Pi}$ that involves at most one honest session of each role. Since C is satisfiable, there exists a constraint system D in solved form, a substitution σ, and a natural number n such that: $C \rightsquigarrow^n_\sigma D$.

Let $T = \mathsf{maxlhs}(C) \cup \mathsf{rhs}(C)$. We have that T is a set of k-tagged labeled terms and it is easy to check that the constraint system C is well-formed w.r.t. T. Condition 2 is ensured thanks to Condition 2 of Theorem 1. Thanks to Lemma 1, we have that D is well-formed w.r.t. $T\sigma$ and thus for any sid, we have that $\mathit{tags}(T\sigma, \mathit{sid}) = \mathit{tags}(T, \mathit{sid})\sigma$, and that $|\mathit{tags}(T\sigma, \mathit{sid})| \leq 1$. We distinguish two cases depending on the cardinality of $|\mathit{tags}(T\sigma, \mathit{sid}_h)|$.

Case: $\mathit{tags}(T\sigma, \mathit{sid}_h) = \emptyset$. In such a case, we have in particular that $\tilde{m}$ is a term obtained by applying pairing function symbols over some atomic data and also that there is no encrypted subterm involved in the role $\Pi(r)$. Otherwise $|\mathit{tags}(T\sigma, \mathit{sid}_h)|$ would be greater than or equal to 1. In such a case, this means that there is an attack on $\tilde{\Pi}$ that involves only the session sid_h that is honest. Hence the result.

Case: $|\mathit{tags}(T\sigma, \mathit{sid}_h)| = 1$. Let $\mathsf{tag} = \mathit{tags}(T\sigma, \mathit{sid}_h)$ and $S = \{\mathit{sid} \mid \mathit{tags}(T\sigma, \mathit{sid}) = \mathsf{tag}\}$. Let $\mathsf{CK} = \mathit{lgKeys}(C) \setminus T_0$. The two additional conditions required to apply Lemma 2 are satisfied (thanks in particular to our Condition 1 in Theorem 1). From Lemma 2 we know that there exists a natural number $m \leq n$ such that $C|_S \rightsquigarrow^m_{\sigma'} D|_S$ where $\sigma' = \sigma|_X$ and $X = \bigcup_{\mathit{sid} \in S} \mathit{vars}(C, \mathit{sid})$. Since D is in solved form so is $D|_S$; hence, $C|_S$ is satisfiable. Note that $C|_S$ corresponds to the constraint system associated to $\mathsf{sc}' = \mathsf{sc}|_S$ (i.e. the subsequence of sc where we only keep sessions in S) and $\alpha' = \alpha|_S$. Thus, this means that there is an attack on $\tilde{m}$ involving only sessions that are in S.

What it remains to show is that S contains at most one honest session of each role. In order to achieve this, we prove that two distinct sessions sid_1 and sid_2 that are in S and that execute the same role r cannot be tagged in the same way in $T\sigma'$, i.e. $tags(T\sigma', sid_1) \neq tags(T\sigma', sid_2)$. Let $sid_1,\ sid_2 \in S$ be two distinct sessions of the same role r ($1 \leq r \leq k$). By Lemma 1 and the fact that $sid_1, sid_2 \in S$, we deduce that:

- $tags(T\sigma', sid_1) = (tags(T, sid_1))\sigma'$ and $tags(T\sigma', sid_2) = (tags(T, sid_2))\sigma'$,
- $|tags(T\sigma', sid_1)| = 1$ and $|tags(T\sigma', sid_2)| = 1$.

By definition of σ', for any $sid \in S$, we have $tags(T\sigma', sid) = tags(T\sigma, sid)$. Thus, we have that $tags(T\sigma', sid_1) = tags(T\sigma', sid_2) = \mathsf{tag}$. For $\ell = 1, 2$, we have also that $tags(T\sigma', sid_\ell) = \langle \tau_1^\ell, \ldots, \tau_k^\ell \rangle$ where $\tau_i^\ell = \langle a_i^\ell, u_i^\ell \rangle$ for some agent $a_i^\ell \in \mathcal{A}$ and some term u_i^ℓ. Moreover, we have that u_r^1 and u_r^2 are actually two distinct nonces since they have been generated by two different sessions of the role r. Thus, we have that $u_r^1 \neq u_r^2$ and the two distinct sessions sid_1 and sid_2 of the same role r can not have the same tag. Thus they can not be both in S.

Lastly, we have to ensure that the sessions in S are honest. By definition of secrecy, we know that sid_h is an honest session. Thus, we have that $\mathsf{tag} = tags(T\sigma, sid_h) = tags(T\sigma', sid_h) = \langle \tau_1^h, \ldots, \tau_k^h \rangle$ where $\tau_i^h = \langle a_i^h, u_i^h$ and $a_i^h \in \mathcal{A} \setminus \{\epsilon\}$. Thus, since sessions that are put in S are those such that $tags(T\sigma, sid) = tags(T\sigma', sid) = \langle \tau_1^h, \ldots, \tau_k^h \rangle$ and thus those such that $tags(T, sid)\sigma = tags(T, sid)\sigma' = \langle \tau_1^h, \ldots, \tau_k^h \rangle$ where the first component of each τ_i^h is an honest agent. This implies since variables representing participants are instantiated from the beginning that sessions in S are honest sessions. Hence the result.

At this point, we have shown that if $\tilde{\Pi}$ admits an attack, then $\tilde{\Pi}$ admits an attack involving at most one honest session of each role. To conclude, it remains to show that this attack also exist in Π (i.e. the protocol without preamble and without tagging) what is quite obvious. □