Communicating assertions in security protocols: formal models and complexity*

R. Ramanujam[1], Vaishnavi Sundararajan†[2], and S. P. Suresh‡[2]

1 **The Institute of Mathematical Sciences**
Chennai, India
jam@imsc.res.in

2 **Chennai Mathematical Institute**
Chennai, India
{vaishnavi, spsuresh}@cmi.ac.in

Abstract

Cryptographic protocols often have agents sending each other certain facts certifying partial truths about terms – for example, an agent may wish to tell another that an encrypted secret was actually chosen from a particular set of values. In protocols based on the Dolev-Yao model, such certification is often modelled via sequences of communication, or by introducing new cryptographic primitives. We suggest that a separation of terms and assertions helps towards simplifying the logical analysis of such protocols. We work with a simple assertion language using disjunction (to model partial information certificates), and a *says* modality (to model assertions forwarded by agents to others). We study the complexity of the derivability problem, and also study the assertion language when augmented with other reasonable constructs.

Keywords and phrases Security protocols, Dolev-Yao, assertions, intruder deduction

1 Introduction

1.1 Certificates in security protocols

A fairly common feature of cryptographic protocols is the communication of facts about terms occurring in the protocol. Protocols involving interactive proofs, for example, often require agents to certify that the terms they send to other agents possess certain properties [21]. Yet another scenario that illustrates the use of such certification, or *assertions* is in e-voting protocols [8, 16, 20, 26]. Consider the following situation, which is quite common in voting. Agent A encrypts her vote v and sends it to B, along with an assertion that the vote is valid, i.e. v takes on one of a few (pre-specified) values. It should not be possible for B to forward this encrypted vote to someone else, and convince the receiver of its being valid, since a proof of validity might (and almost always will) require B to have access to the vote v (refer to [32] for more examples of this kind). Commonly used protocols which employ certificates which may only be verified but not forwarded include TLS and HTTPS, and therefore a robust and effective way of formally modelling such certification is necessary.

There are many well-established ways to model security protocols in an abstract manner. Possibly the most famous among these is the Dolev-Yao model [17]. This model views terms as elements in a term algebra, and allows encryption and pairing as basic operations to create new terms via a standard set of derivation rules (given in Figure 1). The intruder can see any messages that

* This work is a revised and expanded version of [28].
† Supported by a TCS Research Fellowship, and partially by a grant from the Infosys Foundation.
‡ Partially supported by a grant from the Infosys Foundation.

are sent on the channel, and pass along messages it can generate in any agent's name, but cannot break encryption – this is the *perfect encryption assumption*. In the real world, where encryption is often randomized or based on public-key infrastructure, but private channels are cost-intensive, this is a reasonable assumption of the intruder's abilities. In the thirty years since it was proposed, the Dolev-Yao model has been extended with various cryptographic operations like homomorphic encryption, blind signatures etc. [26, 8, 16] which are relevant to the protocol's application area – and therefore it does not require a great stretch of imagination to believe that there might be a way to imbue this model with the power of certification.

However, one important aspect of the Dolev-Yao model (which is preserved across all these extensions with cryptographic operations) is that agents 'own' any messages which they have received. Suppose agent A sends to B a term $\{t\}_k$ where k is not known to B. The Dolev-Yao model allows B to forward this term to C in its own name, even though B has no idea what t is. Thus, the model treats terms as tokens that can be copied and passed along by the recipients in their own names. Clearly this model will not work with certificates, where we wish to preserve origin information.

$$\frac{}{X \cup \{t\} \vdash t}\ ax$$

$\dfrac{X \vdash t_1 \quad X \vdash t_2}{X \vdash (t_1, t_2)}\ pair$	$\dfrac{X \vdash (t_1, t_2)}{X \vdash t_i}\ split$
$\dfrac{X \vdash t \quad X \vdash k}{X \vdash \{t\}_k}\ enc$	$\dfrac{X \vdash \{t\}_k \quad X \vdash inv(k)}{X \vdash t}\ dec$

Figure 1 The Dolev-Yao derivation system

Here are a few ways in which the Dolev-Yao model is modified to handle certification.

- In some simple cases, one need not explicitly model assertions in the protocol at all, and instead rely on the conventions and features of the framework itself. Examples would be to certify the goodness of keys, freshness of nonces etc [11].
- In other cases, one uses cryptographic devices like *zero knowledge proofs* [21], *bit-commitment schemes* [27, 6, 25] etc. to make partial secrecy assertions. For example, a voter V might post an encrypted vote $\{v\}_k$ along with a zero knowledge proof that v is either 0 or 1. This allows an authority to check that the voter and the vote are legitimate, without needing to know the actual value of the vote.
- For some assertions, one could use ad hoc conventions specific to a protocol. For instance, a term that models an assertion α might be paired (or tagged) with the agent name S to signify that S generated (and therefore owns) the assertion α [11].

However, it often proves to be the case that ad hoc methods do not hold up well under composition of protocols. Zero knowledge proofs or bit-commitment schemes are implemented as multiple sequences of communicated messages, and are therefore hard to 'see' as assertions of the facts that they are certifying. Neither of these methods, therefore, allows us to structure the protocol in a readable manner.

1.2 Our model

We propose a new extension to the Dolev-Yao model in which agents have the ability to communicate assertions explicitly, rather than via any encoding. The fact that certification can be expressed in the Dolev-Yao model via various methods of translation suggests that this ability does not add any expressive power. However, treating data and assertions as different abstract objects gives significant flexibility for formal specification and reasoning about such protocols. In fact, the need for formal methods in security protocols goes beyond verification and includes ways of structuring protocols [1, 2], and a syntactic separation of the term algebra and an associated language of assertions should be seen in this light.

Note that our assertions are in the spirit of [3], where the zk primitive encapsulates a zero-knowledge proof, in that the implementation for the assertion might have a zero-knowledge proof of the fact. However, the abstraction in [3] treats a zk term as part of the term algebra, whereas our model introduces a different algebra altogether whose elements are the assertions. Our assertion language is inspired by the language of *patterns* used in [7], which considers a model where each term is tagged with an abstract pattern to which the term conforms.

Much as the Dolev-Yao model is a minimal term algebra with its derivation rules, we consider the extension with assertions also as a minimal logic with its associated derivation rules. We suggest a propositional modal language, with highly restricted use of negation and modalities, inspired by infon logic [22, 5] (which reasons about access control). The fundamental assertion is one about term structure, which claims that a particular atomic term is part of a larger term, and has the form a *occurs in* t (for a the atomic term occurring in the larger term t). A priori, certification in cryptographic protocols reveals partial information about terms. Therefore, we use disjunction to make the revelations partial. For instance, (0 *occurs in* t)$\vee$(1 *occurs in* t) can be seen as a partial secrecy assertion. Note that background knowledge of the Dolev-Yao model offers implicit atomic negation: 0 *occurs in* $\{m\}_k$ where m is atomic will exclude the assertion 1 *occurs in* $\{m\}_k$. With conjunctions to bundle assertions together, we have a restricted propositional language. Communications are term-assertion pairs where the term or the assertion field could potentially be empty.

The modality we study is one that refers to agents passing assertions along, and has the flavour of delegation: A sending α to B does not allow B to directly send α to other agents, but instead allows B to send the assertion A *says* α to other agents. Many papers which view assertions as terms work with assertions similar to the ones used here. For instance, [9] presents a new cryptographic primitive for partial information transmission, while [19] deals with delegation and signatures, although there the focus is more on anonymity and group signatures.

A natural question would be: how are such assertions to be verified? Should the agent generating the assertion construct a proof and pass it along as well? This is the approach followed in protocols using zero-knowledge proofs [4]. While such an approach is natural in models that view assertions as terms, treating them as distinct allows us an abstraction similar to Dolev-Yao's perfect encryption assumption mentioned earlier. We call it the *reliable assertion assumption*: the model ensures the correctness of assertions at the point of generation, and honest principals assume such correctness of any assertions they might receive and proceed. This paradigm also ensures that in order to 'prove' the correctness of any assertion sent out by it, an agent needs to have access to all the terms the assertion talks about, thereby ruling out situations where B tries to forward in its own name an assertion α sent to it by A. As a design choice, in our model, all assertions sent out by agents are signed by them, and signatures cannot be forged. The intruder, therefore, can replay earlier messages, but not forge new messages in other agents' names.

Which brings us to our next question: is there any difference between the terms appearing in our protocol specification/formalization, and the ones communicated at runtime? In the Dolev-Yao model, the terms appearing in the protocol specification ('abstract' terms) are instantiated at

runtime by means of substitution functions, to yield 'concrete' terms. Therefore, agents have abstract terms that serve as templates, which are mapped to actual terms by substitutions. We extend this idea in our model. The assertions are part of the protocol specification itself, and refer to abstract terms appearing in the protocol. All assertions are therefore instantiated at runtime, and the concrete form of an abstract assertion depends on the concrete terms it references. For example, if we have an assertion of the form a *occurs in* t, and the substitution function for the terms maps a to the atomic term 1, and maps t to the encrypted term $\{1,2\}_k$, then the concrete assertion we obtain at runtime would be be 1 *occurs in* $\{1,2\}_k$.

Note that this allows agents to send totally unrelated terms and assertions together (as long as the terms mentioned in the assertions have been communicated earlier) – for example, an agent could send a communication of the form $\{1\}_{k(A,B)}, \{2 \textit{ occurs in } \{(2,3)\}_{k(A,S)}\}$ where the assertion has no connection to the term that is sent with it. The interaction between an assertion and the term sent along with it can be varied, in order to obtain better abstractions for the model (and potentially increase expressivity). One way to augment our existing model, thereby increasing term-assertion interaction, is to introduce a placeholder $\diamondsuit$, and a new 'open' assertion of the form a *occurs in* $\diamondsuit$, which is always accompanied by a term. In order to obtain the concrete form of this assertion at runtime, the placeholder is replaced by the accompanying term, and then the uniform substitution function applied to all terms.

1.3 Examples of certificates

We now informally present a few examples illustrating how certification is used in security protocols and how our basic model will capture this. We also present some attacks where the malicious intruder uses the fact that assertions are transmitted, and tries to gain more information than she is entitled to.

▶ **Example 1.** Consider a very simple scenario where A sends to B a term m encrypted in A's public key, about which he only wants B to know that the encrypted term is one of two possible values a and b. Since B cannot decrypt the term, the assertion is the only way B learns anything about the term that is encrypted.

$$A \to B : \{m\}_{pk(A)}, \{(m \textit{ is } a) \vee (m \textit{ is } b)\}$$

▶ **Example 2.** This example is a slight variation on the earlier scenario. Now A sends to B a pair (m,n) encrypted in A's public key, about which he only wants B to know that somewhere in the encrypted term occurs a nonce which is either a or b. Again, since B cannot decrypt the term, the assertion is the only way B learns anything about the term that is encrypted. But note that this is a weaker assertion than earlier, since B does not learn exactly where in the term a or b appears.

$$A \to B : \{(m,n)\}_{pk(A)}, \left\{(a \textit{ occurs in } \{(m,n)\}_{pk(A)}) \vee (b \textit{ occurs in } \{(m,n)\}_{pk(A)})\right\}$$

▶ **Example 3.** Here we present a toy e-voting protocol. Agent A is a voter, talking to an authority T. T needs to be convinced by A that A is indeed who he says he is (i.e. A), and that his vote is valid (i.e. one of the two acceptable choices a and b). Based on A's vote, T will also update the tally for the candidate A votes for. A therefore sends to T a pair composed of A's identity, and his vote v_A, encrypted in the key A shares with T. T decrypts the message, checks the vote to see if it is one of the acceptable values, and returns a confirmation c_A to A if it is true, otherwise sends 0.

$$A \to T : (A, v_A)\}_{k(A,T)}$$

If T confirms $\{(v_A = a) \vee (v_A = b)\}$ $T \to A : c_A$. Otherwise $T \to A : 0$.

▶ **Example 4.** Now consider a slightly more involved voting protocol. A is a voter, but now we have an administrator T and a counter C. T needs to be convinced by A that A is indeed A, and that his vote is for either a or b. However, T should not be able to see A's exact vote, which should be seen only by the counter C, who will update the tally for the person A votes for, based on A's vote. In this example, A sends to the admin a pair encrypted in the key he shares with T. The pair contains A's identity, and A's vote encrypted in the key A shares with C (which T cannot decrypt, and must pass on as is to C). A also sends an accompanying assertion to state that his vote is indeed for one of the preset candidates. T decrypts the message, ensures that the identity is indeed that of A, and sends the encrypted vote to C, along with the assertion that came from A. C checks the vote, updates the tally, and issues a confirmation to A if the vote has been received, and the assertion validated. Otherwise C sends a o. Note that T needs to send this assertion to C mentioning that A sent it to T earlier (as opposed to T having generated this assertion), and therefore must use an A *says* prefix.

$$A \to T : \{(A, \{v_A\}_{k(A,C)})\}_{k(A,T)},\ \{(v_A \text{ is } a) \vee (v_A \text{ is } b)\}$$

$$T \to C : \{v_A\}_{k(A,C)},\ \{A \textit{ says } (v_A \text{ is } a) \vee (v_A \text{ is } b)\}$$

If C confirms that v_A is a or b , $C \to A : c_A$. Otherwise $C \to A : 0$.

2 Model

In this section, we present the various features of our model in detail.

2.1 The term model

Fix countable sets Ag, $\mathcal{N}$ and $\mathcal{K}$, denoting the set of *agents*, *nonces* and *keys*, respectively. The set of *basic terms* is $\mathcal{B} = Ag \cup \mathcal{N} \cup \mathcal{K}$. For each $A, B \in Ag$, assume that $sk(A)$ and $pk(A)$ are private-public key pairs, and $k(A, B)$ are shared keys. Further, each $k \in \mathcal{K}$ has an *inverse* defined as follows: $inv(pk(A)) = sk(A)$, $inv(sk(A)) = pk(A)$ and $inv(k) = k$ for the other keys. The set $\mathcal{T}$ of Dolev-Yao terms is given by the following syntax (where $m \in \mathcal{B}$ and $k \in \mathcal{K}$)[1]:

$$t := m \mid (t_1, t_2) \mid \{t\}_k$$

The most important aspect of the Dolev-Yao model is the system of rules that govern what new terms can be constructed from the set of terms already known to an agent. This is usually presented in the form of a proof system involving *sequents* of the form $X \vdash t$, where $X \subseteq_{\text{fin}} \mathcal{T}$ and $t \in \mathcal{T}$. The proof system is given in Figure 1. We use the notation $X \vdash_{dy} t$ to denote that there is a proof of $X \vdash t$ according to the Dolev-Yao proof rules, and $X \vdash_{dy} T$ (for a set of terms T) to denote that $X \vdash_{dy} t$ for all $t \in T$.

We use $st(t)$ to denote the set of *subterms* of t and $st(X) = \bigcup_{t \in X} st(t)$ in Proposition 5, which is a well-known fact about the basic Dolev-Yao model [33].

▶ **Proposition 5.** *Given $X \subseteq_{fin} \mathcal{T}$ and $t \in \mathcal{T}$, it can be decided whether $X \vdash_{dy} t$ in time linear in $|st(X \cup \{t\})|$.*

[1] We keep our term model simple by allowing only *atomic keys*. One can enrich the syntax by allowing *constructed keys*, which would admit terms of the form $\{m\}_{(n,p)}$, $\{m\}_{\{n\}_p}$, etc. Our work can adapted to the case of constructed keys without much difficulty.

2.2 The assertion language

The set of assertions, $\mathscr{A}$, is given by the following syntax:

$$\alpha := m \prec t \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2 \mid A \textit{ says } \alpha$$

where $m \in \mathscr{B}$, $A \in Ag$ and $t \in \mathscr{T}$. The set of *subformulas* of a formula α is denoted $sf(\alpha)$. The assertion $m \prec t$ is to be read as "m occurs in t". Note that when the term communicated is an encrypted nonce $\{a\}_k$, the assertion $m \prec t$ states that m is a (as in our examples presented earlier).

The *says* modality captures the flavour of delegation. An agent B, upon sending agent C the assertion A *says* α, conveys to C that he has obtained this assertion α from the agent A, and that while he has no proof of α, A hopefully does. A has, in essence, allowed him to transmit this assertion to other agents.

Our syntax is very modest – there is no negation or implication, and the modalities are $\Box$-like. While general negation and implication is important in the context of access control and delegation, and it is interesting to explore their use in the context of our work, those operators also brings with them increased complexity. While the *says* modality is natural in the context of delegation and is naturally $\Box$-like, it is hard to come up with a natural modality dual to *says* in our context. Integrating more operators to our syntax in an appropriate manner is work for the future.

The proof rules for assertions are presented using *sequents* of the form $X, \Phi \vdash \alpha$, where X and Φ are finite sets of terms and assertions respectively, and α is an assertion. For ease of presentation, we present the rules in three parts.

$\dfrac{}{X, \Phi \cup \{\alpha\} \vdash \alpha}\, ax_1$	$\dfrac{X, \Phi \vdash m \prec \{b\}_k \quad X, \Phi \vdash n \prec \{b\}_k}{X, \Phi \vdash \alpha} \perp (m \neq n;\ b \in \mathscr{B})$
$\dfrac{X, \Phi \vdash \alpha_1 \quad X, \Phi \vdash \alpha_2}{X, \Phi \vdash \alpha_1 \wedge \alpha_2} \wedge i$	$\dfrac{X, \Phi \vdash \alpha_1 \wedge \alpha_2}{X, \Phi \vdash \alpha_i} \wedge e$
$\dfrac{X, \Phi \vdash \alpha_i}{X, \Phi \vdash \alpha_1 \vee \alpha_2} \vee i$	$\dfrac{X, \Phi \vdash \alpha_1 \vee \alpha_2 \quad X, \Phi \cup \{\alpha_1\} \vdash \beta \quad X, \Phi \cup \{\alpha_2\} \vdash \beta}{X, \Phi \vdash \beta} \vee e$

Figure 2 The rules for deriving assertions: propositional fragment

The proof system has appropriate rules for conjunction and disjunction, as presented in Figure 2. The contradiction rule merits some discussion. We employ a restricted contradiction rule $\perp$ which captures contradiction at the atomic level (whereby two assertions claiming that different nonces appear in an encrypted nonce are deemed contradictory), and from a contradiction one can derive anything. Some care is needed in deciding which pairs of assertions we deem to be contradictory. For instance, if t is a pair, we cannot say that $m \prec t$ and $n \prec t$ are contradictory, since t could very well be (m, n). So the term t should either be atomic, or be built using encryptions only.

We next present the rules for atomic assertions of the form $m \prec t$ in Figure 3. We wish to construct assertions about compound terms using assertions for (some of) the constituent terms. To this end, we consider assertions corresponding to compound terms formed using the operations

allowed by the Dolev-Yao theory, namely encryption, decryption, split and pairing. Note that all these rules require X to be nonempty, and refer to derivations in the Dolev-Yao theory. For an agent to derive an assertion about a term t, it should be capable of constructing t from the ground up, which is modelled by saying that from X one can learn (in the Dolev-Yao theory) all basic terms occurring in t (including all the encryption keys occurring in t). For example, in the *split* rule, suppose the agent can derive from X all of $st(t_i \cap \mathscr{B})$, and that m is not a basic term in t_i. The agent can now derive $m \prec t_{1-i}$ from $m \prec (t_0, t_1)$.

$$\frac{X \vdash_{dy} m}{X, \Phi \vdash m \prec m}\, ax_2$$

$$\frac{X \vdash_{dy} \{t\}_k \quad X \vdash_{dy} k \quad X, \Phi \vdash m \prec t}{X, \Phi \vdash m \prec \{t\}_k}\, enc \qquad \frac{X \vdash_{dy} inv(k) \quad X, \Phi \vdash m \prec \{t\}_k}{X, \Phi \vdash m \prec t}\, dec$$

$$\frac{X \vdash_{dy} (t_0, t_1) \quad X, \Phi \vdash m \prec t_i \quad X \vdash_{dy} st(t_{1-i}) \cap \mathscr{B}}{X, \Phi \vdash m \prec (t_0, t_1)}\, pair$$

$$\frac{X, \Phi \vdash m \prec (t_0, t_1) \quad X \vdash_{dy} st(t_i) \cap \mathscr{B} \quad m \notin st(t_i)}{X, \Phi \vdash m \prec t_{1-i}}\, split$$

Figure 3 The rules for atomic assertions

$$\frac{X, \Phi \vdash \sigma : (m \prec \{b\}_k) \quad X, \Phi \vdash \sigma : (n \prec \{b\}_k)}{X, \Phi \vdash \sigma : \alpha}\, \bot\, (m \neq n)$$

$$\frac{X, \Phi \vdash \sigma : \alpha_1 \quad X, \Phi \vdash \sigma : \alpha_2}{X, \Phi \vdash \sigma : (\alpha_1 \wedge \alpha_2)}\, \wedge i \qquad \frac{X, \Phi \vdash \sigma : (\alpha_1 \wedge \alpha_2)}{X, \Phi \vdash \sigma : \alpha_i}\, \wedge e \qquad \frac{X, \Phi \vdash \sigma : \alpha_i}{X, \Phi \vdash \sigma : (\alpha_1 \vee \alpha_2)}\, \vee i$$

$$\frac{X, \Phi \vdash \sigma : (\alpha_1 \vee \alpha_2) \quad X, \Phi \cup \{\sigma : \alpha_1\} \vdash \sigma : \beta \quad X, \Phi \cup \{\sigma : \alpha_2\} \vdash \sigma : \beta}{X, \Phi \vdash \sigma : \beta}\, \vee e$$

Figure 4 The rules for *says* assertions

Figure 4 gives rules for assertions of the form A *says* α. For $\sigma = A_1 A_2 \cdots A_n$, $\sigma : \alpha$ denotes A_1 *says* $(A_2$ *says* $\cdots (A_n$ *says* $\alpha) \cdots)$, and $\sigma : \Phi = \{\sigma : \alpha \mid \alpha \in \Phi\}$. These rules are direct generalizations of the propositional rules in Figure 2, and permit propositional reasoning in a *fixed* modal context. This is captured by the fact that the same σ occurs in the premises and conclusion of the rules.

We denote by $X, \Phi \vdash_{alp} \alpha$ (resp. $X, \Phi \vdash_{alat} \alpha$; $X, \Phi \vdash_{als} \alpha$; $X, \Phi \vdash_{al} \alpha$) the fact that there is a

derivation of $X, \Phi \vdash \alpha$ using the rules in Figure 2 (resp. ax_1 and the rules in Figure 3; ax_1 and the rules in Figure 4; the rules in Figures 2, 3 and 4).

Note that our rules for assertions allows only propositional reasoning in the context of a (fixed) σ. We can obtain an equivalent proof system by replacing all the rules in Figure 4 by the following rule.

$$\frac{\varnothing, \Phi \vdash \alpha}{\varnothing, \sigma : \Phi \vdash \sigma : \alpha}$$

But the system in Figure 4 has the structural advantage that the set of formulas to the left of $\vdash$ is the same in the premises and conclusion in all the rules other than $\vee e$. This helps in the presentation of some of the algorithms in Section 4.

2.3 Active intruder model

Protocols are usually specified as multiple sequences of communication between agents. Given below is a typical specification of a simple protocol, where A wants to check if B is online. A sends out a fresh nonce m encrypted in B's public key, and waits for B to tell her that he is online by responding with the same nonce encrypted in A's public key. Let us call this protocol $\mathfrak{P}$.

$$\left.\begin{array}{l} A \to B : \{m\}_{pk(B)} \\ B \to A : \{m\}_{pk(A)} \end{array}\right\} \textbf{Protocol } \mathfrak{P}$$

However, towards simplifying the task of formal analysis, we shall consider a different specification of protocols. Note that this new specification can be generated from the above kind of specification.

All protocols admit *send* and *receive* actions. Consider the first communication in $\mathfrak{P}$. This is broken into two actions – a send and a receive.

$A!B : (M)\{m\}_{pk(B)}$ (A sends out to intended recipient B)

$B?A : \{m\}_{pk(B)}$ (B receives from purported sender A)

M is the set of nonces of the term $\{m\}_{pk(B)}$ that ought to be fresh, i.e. regenerated for each new session (in our case $M = \{m\}$, but we will write $A!B : (m)\{m\}_{pk(B)}$ instead of $A!B : (\{m\})\{m\}_{pk(B)}$). Note that the intruder might intercept messages, so not all sends need have corresponding receives (which is why this break-up of a single communication into two actions at the two agents' ends is necessary). Any receive by any agent is implicitly a send by the intruder, since the intruder always snoops on the channel.

▶ **Definition 6** (Role)**.** A role for $A \in Ag$ (denoted by A-role) is a finite sequence of actions $a_1 \cdots a_k$ where each a_i is a send (sender A), receive (receiver A), confirm or deny action by A. A role is therefore an A-role for some agent A.

Breaking up all communications into actions, we get A's role in $\mathfrak{P}$ to be the following sequence.

$A!B : (m)\{m\}_{pk(B)}$

$A?B : \{m\}_{pk(A)}$

The corresponding B-role is the following.

$B?A : \{m\}_{pk(B)}$

$B!A : \{m\}_{pk(A)}$

Observe that the m in the message sent by B is the one it received in the earlier message (purportedly) from A, and not fresh. Thus the set of fresh nonces, M, is empty. This is more accurately depicted as $B!A : (\varnothing)\{m\}_{pk(A)}$, but we stick to our simpler notation in this case.

Actions

In addition to the send and receive actions in our model (all assertions involved in these actions are signed by the senders), we also admit two more types of actions, namely *confirm* and *deny*. These capture conditional branching. An agent might, depending on whether he/she can verify some *says* free assertion, perform action a_1 if the condition is true, and action a_2 otherwise. The behaviour of an agent A in a protocol, in the presence of a branch on assertion α, is represented by two different branches, one in which the action at that stage is A : *confirm* α (i.e. α is true, and A confirms α) and the other in which it is A : *deny* α. These actions are illustrated below.

Action type	Notation	
Send from A to B	$A!B : [(M)t, \{\alpha\}_{sd(A)}]$	
Receive by A from B	$A?B : [t, \{\alpha\}_{sd(B)}]$	
Confirm by A	A : *confirm* α	α *says* free
Deny by A	A : *deny* α	

Figure 5 Actions in our system

In the send action, M is the set of fresh nonces used by A in the construction of t. Agents are allowed to send terms and assertions separately (i.e. in the send and receive actions, either the term or the assertion could be omitted, but not both). Note that in our model, we consider branching based only on the assertions that the agent can locally verify, and we do not consider assertions that might have been communicated to the agent by other agents. For confirm and deny actions, therefore, α is *says* free. Examples 3 and 4 illustrate the use of these actions.

▶ **Definition 7** (Protocol). A protocol Pr is a pair $(const, R)$, where $const \subseteq \mathscr{B}$ is the set of constants of Pr, and R is a finite set of roles.

Substitution

Observe that so far everything we have talked about is abstract – agent names, terms, assertions etc. In order to take these abstract objects to concrete entities which are actually communicated during runtime, we use a *substitution*. A substitution σ is a homomorphism on $\mathscr{T}$ ssatisfying $\sigma(Ag) \subseteq Ag$ and $\sigma(\mathscr{K}) \subseteq \mathscr{K}$ (i.e. abstract terms referring to agents' names and keys are always mapped into the set of agents' names and the set of keys respectively). Note that we allow non-atomic substitutions, i.e. a nonce can be instantiated with a complex term at run time, but keys are always mapped to keys (we do not allow constructed keys). σ is said to be suitable for a protocol $Pr = (const, R)$ if $\sigma(m) = m$ for all $m \in const$. For an abstract assertion α, $\sigma(\alpha)$ is the concrete assertion obtained by applying σ to all the abstract terms appearing in α.

Role Instance

A *role instance* of a protocol Pr is a tuple $ri = (\eta, \sigma, lp)$, where η is a role of Pr, σ is a substitution suitable for Pr, and $0 \leqslant lp \leqslant |\eta|$. $ri = (\eta, \sigma, 0)$ is said to be an *initial role instance*. The set of role instances of Pr is denoted by $RI(Pr)$. $IRI(Pr)$ is the set of initial role instances of Pr. For a role instance $ri = (\eta, \sigma, lp)$, $ri + 1 = (\eta, \sigma, lp + 1)$. If $ri = (\eta, \sigma, lp)$, $lp \geqslant 1$ and $\eta = a_1 \cdots a_\ell$, $act(ri) = \sigma(a_{lp})$. For $S, S' \subseteq RI(Pr)$ and $ri \in RI(Pr)$, we say that $S \xrightarrow{ri} S'$ iff $ri \in S$, $ri + 1 \in RI(Pr)$ and $S' = (S \setminus \{ri\}) \cup \{ri + 1\}$.

An important detail to observe here is that *ri* refers to abstract terms appearing in the role η. However, $act(ri)$ is obtained by applying the substitution σ to the appropriate action, and therefore refers to concrete terms. Consequently, our actions on which transitions occur are also concrete actions.

Transition system states

A *knowledge state ks* is a tuple of the form $((X_A, \Phi_A)_{A \in Ag}, h)$, where $X_A \subseteq \mathscr{T}$ (resp. $\Phi_A \subseteq \mathscr{A}$) is the set of terms (resp. assertions) A has accumulated by then. h is a finite set of signed assertions of the form $\{\alpha\}_{sd(A)}$. It is intended to model the *relevant history* – the set of all signed assertions communicated in the run so far. The *initial knowledge state* of a protocol $Pr = (const, R)$ is given by $((X^0_A, \varnothing)_{A \in Ag}, \varnothing)$, where

$$X^0_A = const \cup Ag \cup \{sk(A)\} \cup \{pk(B), k(A, B) \mid B \in Ag\}$$

is the set of terms any agent A has access to at the very beginning of a protocol. Actions cause the update of knowledge states – a changes the system state from *ks* to *ks*′ (denoted by $ks \xrightarrow{a} ks'$), which will be defined shortly.

A *protocol state* of *Pr* is a pair $s = (ks, S)$ where *ks* is a knowledge state of *Pr* and $S \subseteq_{\text{fin}} RI(Pr)$. $s = (ks, S)$ is an *initial protocol state* if *ks* is initial and $S \subseteq IRI(Pr)$. For any two protocol states $s = (ks, S)$ and $s' = (ks', S')$, and an action a, we say that $s \xrightarrow{a} s'$ iff there is $ri \in S$ such that $act(ri + 1) = a$, $S \xrightarrow{ri} S'$, a is enabled at *ks*, and $ks \xrightarrow{a} ks'$. The states, initial states, and transitions defined above induce a transition system on protocol states, denoted $TS(Pr)$. A *run* of a protocol *Pr* is any run of $TS(Pr)$.

Transition system updates

We now need to examine how actions affect the information the agents gain about the terms and assertions communicated during the execution of the protocol. Consider two knowledge states $ks = ((X_A, \Phi_A)_{A \in Ag}, h)$ and $ks' = ((X'_A, \Phi'_A)_{A \in Ag}, h')$, and a belonging to one of the four action types shown in Figure 5. We describe in Figure 6 the conditions under which the action a (mentioning concrete terms) is enabled at a knowledge state *ks*, and then in Figure 7 the conditions under which it will update the system from *ks* to *ks*′ (denoted $ks \xrightarrow{a} ks'$).

Action	Enabling conditions
$a = A!B:[(M)t, \{\alpha\}_{sd(A)}]$	$M \cap X_C = \varnothing$ for all C $X_A \cup M \vdash_{dy} t$ $X_A \cup M, \Phi_A \vdash_{al} \alpha$.
$a = A?B:[t, \{\alpha\}_{sd(B)}]$	$X_I \vdash_{dy} t$ $X_I, \Phi_I \vdash_{al} \alpha$ If $B \neq I, \{\alpha\}_{sd(B)} \in h$.
$a = A : \textit{confirm}\ \alpha$	$X_A, \varnothing \vdash_{al} \alpha$
$a = A : \textit{deny}\ \alpha$	$X_A, \varnothing \nvdash_{al} \alpha$

Figure 6 Enabling conditions for a at *ks*

In the above definitions, the relevant history h is crucially used in the enabledness check for an honest agent receive. In the case where an honest agent A receives an assertion α whose purported

Action	Updates
$a = A!B:[(M)t, \{\alpha\}_{sd(A)}]$	$X'_A = X_A \cup M$ $X'_I = X_I \cup \{t\}$ $\Phi'_P = \Phi_P \cup \{\alpha, A \textit{ says } \alpha\}$ for $P = A, I$ $h' = h \cup \{\{\alpha\}_{sd(A)}\}$.
$a = A?B:[t, \{\alpha\}_{sd(B)}]$	$X'_A = X_A \cup \{t\}$ $\Phi'_A = \Phi_A \cup \{B \textit{ says } \alpha\}$.
$a = A : \textit{confirm } \alpha$	No change
$a = A : \textit{deny } \alpha$	

Figure 7 Updates when $ks \xrightarrow{a} ks'$

sender is $B \neq I$, we actually require that B indeed communicated $\{\alpha\}_{sd(B)}$ sometime in the past. This ensures that the intruder cannot send out assertions in the name of another agent B, unless B sent it out in the past.

The change in the X sets is easily justified by an operational model in which the intruder can snoop on the entire network, but agents are allowed to send only terms which they can generate. We have extended the same idea to assertions as well, but there is the extra complication of signing the assertions. The intruder typically has access only to its own signature, and we do not consider corruption of agents by the intruder here. Thus we posit that the intruder can only *replay* assertions signed by other agents, or that it can generate assertions and sign them in its own name.

As an operational justification for why the honest agents cannot send assertions sent by other agents without attaching the *says* prefix, we can imagine the following concrete model. There is a trusted third party verifier (TTP) that allows assertions to be transmitted at large only after the sender provides a justification to the TTP. This means that an honest agent B who receives an assertion α from A cannot pass it on to others, because the TTP will demand a justification for this, which B cannot provide. The intruder, though, can snoop on the network, so it has the bits that A sent as justification for α to the TTP, and thus gets to store α in its local database.

▶ **Example 8.** Towards illustrating how the transition system works, let us add an assertion to our earlier protocol $\mathfrak{P}$ (our running example in the beginning of this section). Let us call this new protocol $\mathfrak{P}'$.

$$\left.\begin{array}{l} A \to B : \{m\}_{pk(B)}, \{m_1 \prec \{m\}_{pk(B)} \vee m_2 \prec \{m\}_{pk(B)}\} \\ B \to A : \{m\}_{pk(A)} \end{array}\right\} \textbf{Protocol } \mathfrak{P}'$$

As earlier, we can split this up into an A-role ($a_1\, a_2$) and a B-role ($b_1\, b_2$) as follows.

a_1: $A!B : [(m)\{m\}_{pk(B)}, \{m_1 \prec \{m\}_{pk(B)} \vee m_2 \prec \{m\}_{pk(B)}\}_{sd(A)}]$ $\quad a_2$: $A?B : \{m\}_{pk(A)}$

b_1: $B?A : [\{m\}_{pk(B)}, \{m_1 \prec \{m\}_{pk(B)} \vee m_2 \prec \{m\}_{pk(B)}\}_{sd(A)}]$ $\quad b_2$: $B!A : \{m\}_{pk(A)}$

Now consider the system where the only constants of the protocol are 1 and 2. A run of the transition system for $\mathfrak{P}' = (\{1, 2\}, \{a_1\, a_2, b_1\, b_2\})$ is illustrated in Figure 8. The actual action taken for the transition is shown to the left of the transition arrow, while the criteria that enabled said action are shown to the right. The substitution is the same for all agents, i.e. $\sigma = \sigma_A = \sigma_B = \sigma_I$, with $\sigma(m) = \sigma(m_1) = 1$ and $\sigma(m_2) = 2$. Note that only the entities that change are shown in the states – for example, in the transition from s^0 to s^1, the X_B and Φ_B sets undergo no change and therefore are not mentioned.

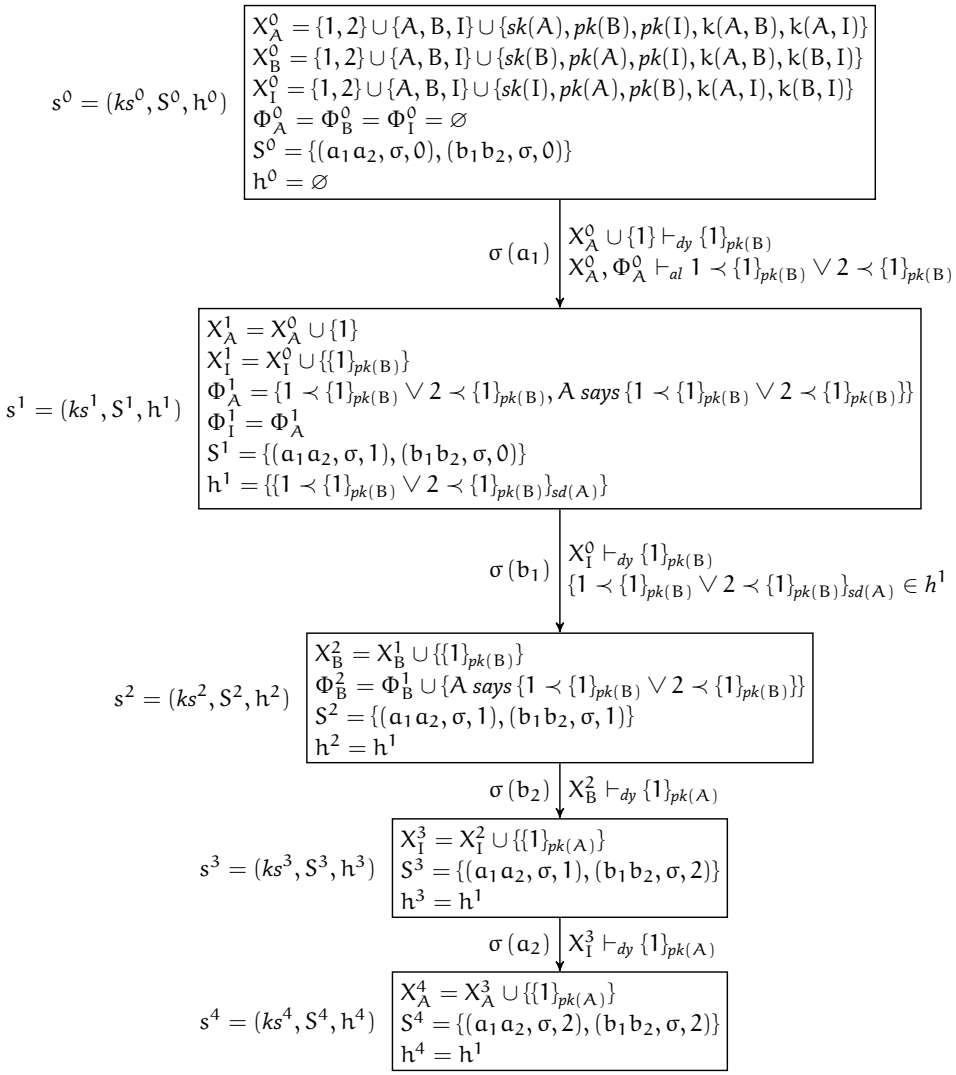

Figure 8 Transition system for Example 8 with $\mathfrak{P}' = (\{1,2\}, \{a_1a_2, b_1b_2\})$

2.4 Derivability Problem

One key aspect of this model is that all the enabling checks are done on *concrete* sets of terms and assertions, and are questions of the form - "Given α, X and Φ, is α derivable from X and Φ?". Since the system needs to appeal to enabling criteria every time a send/receive action occurs, it would be advantageous if such checks could be done efficiently. In Section 4, the complexity of this question will be explored, but first, we present a few examples to illustrate why this problem might be hard (at least in some cases).

▶ **Definition 9** (Derivability Problem)**.** The derivability problem (or the passive intruder deduction problem) is the following: given $X \subseteq \mathscr{T}$, $\Phi \subseteq \mathscr{A}$ and $\alpha \in \mathscr{A}$, determine if $X, \Phi \vdash_{al} \alpha$.

▶ **Example 10.** Suppose we have a protocol where an agent wishes to convince another agent that an atomic term is one of two possible values, without revealing the exact value of said term (recall Example 1 from Section 1.3). So the agent encrypts the term m, and sends out an assertion of the form $(p \prec \{m\}_{pk(A)} \vee q \prec \{m\}_{pk(A)})$.

$$A \to B : \{m\}_{pk(A)}, \{(p \prec \{m\}_{pk(A)} \vee q \prec \{m\}_{pk(A)})\}_{sd(A)}$$

Now, consider a situation where two parallel sessions of this protocol are being carried out by the same agent (say A), where the term is the same across the two sessions. Suppose the term is a, and the sets of values used for the disjunction are $\{1,2\}$ and $\{1,3\}$ across the sessions. The intruder then has access to two assertions: $(1 \prec \{a\}_k \vee 2 \prec \{a\}_k)$, and $(1 \prec \{a\}_k \vee 3 \prec \{a\}_k)$. Let Φ be the set consisting these two assertions. We show that the intruder can learn $1 \prec \{a\}_k$ from Φ, and the derivation for the same is non-trivial. We use the notation φ_i for $i \prec \{a\}_k$, for $i \in \{1,2,3\}$. We also use φ_{12} and φ_{13} to denote $1 \prec \{a\}_k \vee 2 \prec \{a\}_k$ and $1 \prec \{a\}_k \vee 3 \prec \{a\}_k$, respectively.

The intruder's derivation of $\Phi \vdash \varphi_1$ is shown in Figure 9. The contradiction rule is used crucially in this proof, to obtain φ_1 from a set containing both φ_2 and φ_3, which in turn is obtained by performing case analysis on both disjunctions. In this proof it was enough to consider one combination of disjuncts from both assertions. In the next example we shall see a situation where we shall need to consider multiple combinations, which suggests a blow up in terms of proof size.

$$\dfrac{\dfrac{\dfrac{\varphi_{12} \in \Phi}{\Phi \vdash \varphi_{12}}\,ax \qquad \dfrac{}{\Phi, \varphi_1 \vdash \varphi_1}\,ax}{\Phi \vdash \varphi_1}\,\vee e \qquad \dfrac{\dfrac{\dfrac{\varphi_{13} \in \Phi}{\Phi, \varphi_2 \vdash \varphi_{13}}\,ax \qquad \dfrac{}{\Phi, \varphi_2, \varphi_1 \vdash \varphi_1}\,ax}{\Phi, \varphi_2 \vdash \varphi_1}\,\vee e \qquad \dfrac{\dfrac{}{\Phi, \varphi_2, \varphi_3 \vdash \varphi_2}\,ax \qquad \dfrac{}{\Phi, \varphi_2, \varphi_3 \vdash \varphi_3}\,ax}{\Phi, \varphi_2, \varphi_3 \vdash \varphi_1}\,\bot}{\Phi, \varphi_2 \vdash \varphi_1}\,\vee e$$

Figure 9 The intruder's derivation (for Example 10)

▶ **Example 11.** Consider a slight extension of the previous example – now the agent wishes to convince another agent that an atomic term is one of *three* possible values. So the assertion sent out is of the form $(p \prec \{m\}_k \vee q \prec \{m\}_k \vee r \prec \{m\}_k)$. Again consider two sessions of this protocol with the same term a, and the sets of values are $\{1,2,3\}$ and $\{1,4,5\}$. The intruder now has access to assertions $(1 \prec \{a\}_k \vee 2 \prec \{a\}_k \vee 3 \prec \{a\}_k)$ and $(1 \prec \{a\}_k \vee 4 \prec \{a\}_k \vee 5 \prec \{a\}_k)$. Again, as earlier, we show that the intruder can derive $1 \prec \{a\}_k$ from these two assertions.

To save space in displaying the derivation, we adopt the following notation:

- $\varphi_i = i \prec \{a\}_k$, for $i \in \{1, \ldots, 5\}$.

$$\dfrac{\dfrac{\dfrac{\varphi_{145} \in \Phi}{\Phi' \vdash \varphi_{145}}\,ax \qquad \dfrac{}{\Phi', \varphi_1 \vdash \varphi_1}\,ax}{\Phi' \vdash \varphi_1}\,\vee e \qquad \dfrac{\dfrac{\dfrac{\varphi_{45} \in \Phi''}{\Phi'' \vdash \varphi_{45}}\,ax \qquad \dfrac{\dfrac{\varphi_2 \in \Phi''}{\Phi'', \varphi_4 \vdash \varphi_2}\,ax \qquad \dfrac{}{\Phi'', \varphi_4 \vdash \varphi_4}\,ax}{\Phi'', \varphi_4 \vdash \varphi_1}\,\bot \qquad \begin{array}{c}\textit{Similar to } \Phi'', \varphi_4 \vdash \varphi_1 \\ \vdots \\ \Phi'', \varphi_5 \vdash \varphi_1\end{array}}{\Phi', \varphi_{45} \vdash \varphi_1}\,\vee e}{\Phi' \vdash \varphi_1}\,\vee e$$

Figure 10 Proof ϖ_1

$$\dfrac{\dfrac{\varphi_{123} \in \Phi}{\Phi \vdash \varphi_{123}}\,ax \qquad \dfrac{}{\Phi, \varphi_1 \vdash \varphi_1}\,ax \qquad \dfrac{\dfrac{}{\Phi, \varphi_{23} \vdash \varphi_{23}}\,ax \qquad \begin{array}{c}\varpi_1 \\ \vdots \\ \Phi, \varphi_{23}, \varphi_2 \vdash \varphi_1\end{array} \qquad \begin{array}{c}\varpi_2 \\ \vdots \\ \Phi, \varphi_{23}, \varphi_3 \vdash \varphi_1\end{array}}{\Phi, \varphi_{23} \vdash \varphi_1}\,\vee e}{\Phi \vdash \varphi_1}\,\vee e$$

Figure 11 The intruder's derivation (for Example 11)

- $\varphi_{ij} = i \prec \{a\}_k \vee j \prec \{a\}_k$, for $i, j \in \{1, \ldots, 5\}$ and $i < j$.
- $\varphi_{ijl} = i \prec \{a\}_k \vee j \prec \{a\}_k \vee l \prec \{a\}_k$, for $i, j, l \in \{1, \ldots, 5\}$ and $i < j < l$.
- $\Phi = \{\varphi_{123}, \varphi_{145}\}$
- $\Phi' = \Phi \cup \varphi_{23} \cup \{\varphi_2\}$
- $\Phi'' = \Phi' \cup \varphi_{45} = \Phi \cup \varphi_{23} \cup \varphi_{45} \cup \{\varphi_2\}$

Note that whenever we can derive φ_i and φ_j from a set Ψ, for $i \neq j$, we can derive any α using the $\bot$ rule. We will use it to derive a φ_l of our choice. The intruder's derivation (rendered in the above notation) is displayed in Figure 11. This refers to two subproofs, ϖ_1 deriving $\Phi, \varphi_{23}, \varphi_2 \vdash \varphi_1$, and ϖ_2 deriving $\Phi, \varphi_{23}, \varphi_2 \vdash \varphi_1$, respectively. The derivation ϖ_1 is shown in Figure 10, and ϖ_2 can be obtained by replacing φ_3 in place of φ_2 in ϖ_1. Note that ϖ_2 is the same size as ϖ_1.

Inserting these subproofs into the appropriate places in the intruder's derivation, the blow up incurred due to considering all possible combinations of disjuncts on the left hand side is apparent.

3 An extension to the assertion language

In Section 2.2, we presented a basic assertion language. In this section, we proceed to add a different (potentially useful) construct to this existing language, to see how these affect the kinds of certificates agents can provide and the deductive power of the intruder.

In our model so far, even if an agent sends a term t and an assertion α together, it need not be that the assertion α mentions t (α could even be an assertion about a totally different term t'). Sometimes it might be better to have the assertion to be contextually bound to the term sent along with it. For example, suppose we have two terms – $\{(1, 2)\}_k$ and $\{1\}_k$. The same assertion, namely $1 \prec t$ (for t being either of these terms), can be made of both these terms. In our existing model, we would have to regenerate the assertion using the term we are talking about, and end up with two different assertions, even though both essentially claim the same thing (albeit about different terms). The assertion being contextually bound to the term would let us use part of the assertion as a black box, and plug in any term about which the assertion holds into that place. We therefore amend the assertion language slightly to allow the use of such a black box or a placeholder – denoted $\diamondsuit$. Assertions of this form shall be referred to as $\diamondsuit$-open assertions, from now on. Conversely, assertions which do not contain $\diamondsuit$ will be referred to as $\diamondsuit$-closed assertions.

The set of assertions with $\Diamond$, $\mathscr{A}_\Diamond$, is given by the following syntax:

$$\alpha := m \prec t \mid m \prec \Diamond \mid t = t' \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2 \mid A \textit{ says } \alpha$$

where $m \in \mathscr{B}$, $A \in Ag$ and $t, t' \in \mathscr{T}$.

The assertion $m \prec \Diamond$ may be read as "m occurs in this", 'this' being the accompanying term. The assertion may now refer to different terms in different instantiations. For a $\Diamond$-open assertion α accompanied by a term t, $\alpha[\Diamond/t]$ is the $\Diamond$-closed assertion obtained by plugging t in place of the $\Diamond$ in α ($\alpha[\Diamond/t]$ still refers to abstract terms – t in particular – and is concretized at runtime).

As we have seen before, protocol specifications refer to abstract nonces, which are instantiated with different concrete terms in different *sessions* of a protocol run. Formally, the actions that constitute a run are of the form $\sigma(a)$, where σ is a substitution and a is an action in the protocol specification. So we need to define $\sigma(\alpha)$ for an $\alpha \in \mathscr{A}_\Diamond$. The definition is the same whether α is $\Diamond$-open or $\Diamond$-closed – each abstract term r occurring in α is replaced by $\sigma(r)$ (and $\Diamond$ is left untouched). This means that we communicate pairs of the form (t, α) in a protocol run, where α is a $\Diamond$-open assertion. containing $\Diamond$ in a protocol run. However, to check whether the communication is enabled at that point in the run, we check whether $\alpha[\Diamond/t]$ is derived by the sender.

Protocol and role descriptions stay the same as earlier. A knowledge state is now of the form $ks = ((X_A, \Phi_A)_{A \in Ag}, h)$, where X_A is a set of terms, Φ_A is a set of $\Diamond$-closed assertions, h is a set of pairs of the form $(t, \{\alpha\}_{sd(A)})$, where α is an assertion from $\mathscr{A}_\Diamond$ and t is its corresponding term.

Protocol and role descriptions, knowledge states, role instances, and protocol states are defined as earlier. $act(ri)$ for a role instance is also defined as before (except for the communicated assertion having occurrences of $\Diamond$). As before, for two protocol states $s = (ks, S)$ and $s' = (ks', S')$, and an action a, we say that $s \xrightarrow{a} s'$ iff there is $ri \in S$ such that $act(ri + 1) = a$, $S \xrightarrow{ri} S'$, a is enabled at ks, and $ks \xrightarrow{a} ks'$. We now describe the conditions under which a is enabled at ks and $ks \xrightarrow{a} ks'$.

Action	Enabling conditions
$a = A!B : [(M)t, \{\alpha\}_{sd(A)}]$	$M \cap X_C = \varnothing$ for all C $X_A \cup M \vdash_{dy} t$ $X_A \cup M, \Phi_A \vdash_{al} \alpha[\Diamond/t]$.
$a = A?B : [t, \{\alpha\}_{sd(B)}]$	$X_I \vdash_{dy} t$ $X_I, \Phi_I \vdash \alpha[\Diamond/t]$ If $B \neq I$, $(t, \{\alpha\}_{sd(B)}) \in h$.

Figure 12 Enabling conditions for a at ks

▶ **Example 12.** Recall Example 10 discussed in Section 2.4. We present the same again here, but the assertion now uses the $\Diamond$ quantifier.

$$A \to B : \{m\}_{pk(A)}, \{(p \prec \Diamond) \vee (q \prec \Diamond)\}_{sd(A)}$$

The assertion is now more succinct. This does not eliminate the attack illustrated earlier, but it does not introduce new attacks either.

While the extended assertion language with $\Diamond$ seems like a mere syntactic convenience, and the transition rules are a straightforward adaptation from the earlier ones, there are some hidden subtleties. For instance, it is possible that A sends to B a communication of the form

$$\{m\}_k, (0 \prec \Diamond) \vee (1 \prec \{m'\}_k),$$

Action	Updates
$a = A!B:[(M)t, \{\alpha\}_{sd(A)}]$	$X'_A = X_A \cup M$ $X'_I = X_I \cup \{t\}$ $\Phi'_P = \Phi_P \cup \{\alpha[\Diamond/t], A \textit{ says } \alpha[\Diamond/t]\}$ for $P \in \{A, I\}$ $h' = h \cup \{(t, \{\alpha\}_{sd(A)})\}$.
$a = A?B:[t, \{\alpha\}_{sd(B)}]$	$X'_A = X_A \cup \{t\}$ $\Phi'_A = \Phi_A \cup \{B \textit{ says } \alpha[\Diamond/t]\}$.

Figure 13 Updates when $ks \xrightarrow{a} ks'$

which B can forward to C as follows:

$$\{m'\}_k, A \textit{ says } ((0 \prec \{m\}_k) \vee (1 \prec \Diamond)).$$

While this assertion is not *false* per se, this is not the exact syntactic form in A's original communication. We can complicate the definitions to prevent this phenomenon, but more work needs to be done before deciding whether such effects should be allowed or not. Another interesting aspect of the model is the relevant history which is part of the knowledge states. It is a set of pairs of the form $(t, \{\alpha\}_{sd(A)})$. So we store some information about *bindings* (the associations between terms and assertions). A consequence is that the intruder cannot send a communication $t, \{\alpha\}_{sd(B)}$ in the name of another agent B unless B has sent it in the past. Instead, if the history only stores the communicated assertions $\{\alpha\}_{sd(A)}$ without the accompanying terms, then it is possible for the intruder to insert false assertions in a run, by pairing an term with an unrelated assertion communicated in the past. It is interesting to study in more detail the connection between the amount of information about the bindings stored and the power of the intruder.

4 Complexity of the derivability problem

We have spent a lot of effort detailing the elements of our model. Now we turn to some basic algorithmic questions related to it. The fundamental problem of interest is the *intruder deduction problem*: given a protocol *Pr* and a secret $m \in \mathcal{N}$, is there a run of *Pr* in which the intruder learns m? This involves a search for a sequence of actions admissible by the transition system of *Pr*. This problem is known to be undecidable in its full generality, for even the basic Dolev-Yao model [18, 24]. Researchers have obtained decidability for a number of special cases over the years [33, 30, 31], and many tools have also been built which tackle large classes of protocols successfully [15, 10]. The typical approach is to place bounds on various parameters which lead to undecidability. The complexity of the intruder deduction problem has also received much attention in the past [14, 12, 13].

A fundamental ingredient of the intruder deduction problem is the question of whether a given sequence of (concrete) actions is an admissible run of a given protocol. The crucial part of this question is to check whether a term or assertion is derivable from a set of terms and assertions. The derivability of terms is part of the basic Dolev-Yao model itself, and the problem of whether $X \vdash_{dy} t$ is known to be solvable in time $O(|X| + |t|)$. Thus we focus on the following problem in this section.

▶ **Definition 13** (Derivability problem). Given $X \subseteq_{\text{fin}} \mathcal{T}$, $\Phi \subseteq_{\text{fin}} \mathcal{A}$, $\alpha \in \mathcal{A}$, does $X, \Phi \vdash_{al} \alpha$ hold?

We first show that the problem is co-NP-hard, and then provide a simple PSPACE decision procedure. We then provide some optimizations of the decision algorithm.

4.1 Properties of the proof system

The following is a useful property that will be crucially used in the lower bound proof, and is proved in Appendix A.

▶ **Proposition 14.** $X, \Phi \cup \{\alpha \vee \beta\} \vdash_{al} \delta$ *iff* $X, \Phi \cup \{\alpha\} \vdash_{al} \delta$ *and* $X, \Phi \cup \{\beta\} \vdash_{al} \delta$.

Among the rules, ax_1, *split*, *dec* and $\wedge e$ are the *pure elimination rules*, $\vee e$ and $\bot$ are the *hybrid rules*, and the rest (ax_2, *pair*, *enc*, $\wedge i$ and $\vee i$) are the *pure introduction rules*. The premises of a pure elimination rule or a hybrid rule are classified into *major premises* and *minor premises*. For the *split*, *dec* and $\wedge e$ rules, there is only one premise of the form $X, \Phi \vdash \alpha$ and that is its major premise. For the $\vee e$ rule, the the major premise is the sequent $X, \Phi \vdash \sigma : (\alpha \vee \beta)$, involving the disjunction that is eliminated. For the $\bot$ rule, both premises are major.

A *normal derivation* is one satisfying the following conditions:

- The major premise of every pure elimination rule or the $\vee e$ rule (occurring in the proof) is the conclusion of a pure elimination rule.
- The conclusion of any instance of the $\bot$ rule is not the premise of any introduction rule.
- The conclusion of any instance of the $\bot$ rule is not the major premise of any hybrid or elimination rule.

The following fundamental theorem is on standard lines. The proof is found in Appendix A.

▶ **Theorem 15.** *If there is a derivation of* $X, \Phi \vdash \alpha$ *then there is a normal derivation of* $X, \Phi \vdash \alpha$.

The following theorem (whose proof is given in Appendix A) is crucial for our algorithms to work because they provide a bound on the set of formulas we need to work with when we check whether α is derivable from X, Φ. $sf(X, \Phi)$ is the set of all subformulas of formulas in Φ (in the context of the set of terms X), and $st(X, \Phi)$ is the set of subterms of all terms in X and occurring in any $\alpha \in \Phi$. The definitions will be made precise in the Appendix, where we prove the theorem. All we need for the algorithms is that both $sf(X, \Phi)$ and $st(X, \Phi)$ are of size $O(N^3)$, where N is the size of X, Φ.

▶ **Theorem 16** (Subformula property)**.** *Let* π *be a normal derivation with conclusion* $X, \Phi \vdash \alpha$ *and last rule* r. *Let* $X, \Phi' \vdash \beta$ *and* $X \vdash_{dy} u$ *occur in* π. *Then* $\Phi' \subseteq sf(\Phi)$, $\beta \in sf(X, \Phi \cup \{\alpha\})$ *and* $u \in st(X, \Phi \cup \{\alpha\})$. *Furthermore, if* r *is a pure elimination rule, then* $\beta \in sf(X, \Phi)$ *and* $u \in st(X, \Phi)$.

4.2 Lower bound

The hardness result is obtained by reducing the validity problem for propositional logic to the derivability problem. In fact, it suffices to consider the validity problem for propositional formulas in disjunctive normal form for our reduction. We show how to define for each formula φ in disjunctive normal form a set of assertions S_φ and an assertion $\overline{\varphi}$ such that $\varnothing, S_\varphi \vdash \overline{\varphi}$ iff φ is a tautology.

Let $\{p_1, p_2, \ldots\}$ be the set of all propositional variables. Fix infinitely many nonces $n_1, n_2, \ldots$ and a key k. We define $\overline{\varphi}$ as follows, by induction.

- $\overline{p_i} = (1 \prec \{n_i\}_k)$
- $\overline{\neg p_i} = (0 \prec \{n_i\}_k)$
- $\overline{\varphi \vee \psi} = \overline{\varphi} \vee \overline{\psi}$
- $\overline{\varphi \wedge \psi} = \overline{\varphi} \wedge \overline{\psi}$

Suppose $\{p_1, \ldots, p_n\}$ is the set of all propositional variables occurring in φ. Then

$$S_\varphi = \{\overline{p_1 \vee \neg p_1}, \ldots, \overline{p_n \vee \neg p_n}\}.$$

▶ **Lemma 17.** $\varnothing, S_\varphi \vdash_{al} \overline{\varphi}$ *iff* φ *is a tautology.*

Proof. For $\nu \subseteq \{p_1, \ldots, p_n\}$, define $S_\nu = \{\overline{p_i} \mid p_i \in \nu\} \cup \{\overline{\neg p_i} \mid p_i \notin \nu\}$. Note that S_ν is a non-contradictory set of atomic assertions.

By repeated appeal to Proposition 14, it is easy to see that $\varnothing, S_\varphi \vdash_{al} \overline{\varphi}$ iff for all valuations ν over $\{p_1, \ldots, p_n\}$, $\varnothing, S_\nu \vdash_{al} \overline{\varphi}$. We now show that $\varnothing, S_\nu \vdash_{al} \overline{\varphi}$ iff $\nu \models \varphi$. The statement of the lemma follows immediately from this.

- We first show by induction on $\psi \in sf(\varphi)$ that $\varnothing, S_\nu \vdash_{al} \overline{\psi}$ whenever $\nu \models \psi$.
  - If $\psi = p_i$ or $\psi = \neg p_i$, then $\varnothing, S_\nu \vdash_{al} \overline{\psi}$ follows from the ax_1 rule.
  - If $\psi = \psi_1 \wedge \psi_2$, then it is the case that $\nu \models \psi_1$ and $\nu \models \psi_2$. But then, by induction hypothesis, $\varnothing, S_\nu \vdash_{al} \overline{\psi_1}$ and $\varnothing, S_\nu \vdash_{al} \overline{\psi_2}$. Hence it follows that $\varnothing, S_\nu \vdash_{al} \overline{\psi_1 \wedge \psi_2}$, by using $\wedge i$.
  - If $\psi = \psi_1 \vee \psi_2$, then it is the case that either $\nu \models \psi_1$ or $\nu \models \psi_2$. But then, by induction hypothesis, $\varnothing, S_\nu \vdash_{al} \overline{\psi_1}$ or $\varnothing, S_\nu \vdash_{al} \overline{\psi_2}$. In either case, by using $\vee i$, it follows that $\varnothing, S_\nu \vdash_{al} \overline{\psi_1 \vee \psi_2}$.
- We now show that if $\varnothing, S_\nu \vdash_{al} \overline{\varphi}$, then $\nu \models \varphi$. Suppose π is a normal proof of $\varnothing, S_\nu \vdash \overline{\phi}$. It follows from the Subformula Property (Theorem 16) that the only rules that can be applied are ax_1, $\wedge i$, $\vee i$, $\wedge e$, $\vee e$ and $\perp$.

  Suppose that there is an occurrence of the $\wedge e$ rule or $\vee e$ rule in π with major premise $\varnothing, S' \vdash \gamma$. We denote by ϖ the subproof with conclusion $S' \vdash \gamma$. Note that ϖ ends in a pure elimination rule, since π is normal and every pure elimination rule and hybrid rule has as its major premise the conclusion of a pure elimination rule. By Theorem 16, we see that $S' \subseteq sf(S_\nu) = S_\nu$ and $\gamma \in sf(S') \subseteq sf(S_\nu)$. But γ is of the form $\alpha \vee \beta$ or $\alpha \wedge \beta$, and this contradicts the fact that S_ν is a set of atomic assertions. Hence the elimination rules cannot occur in π.

  Suppose now that there is an occurrence of the $\perp$ rule in π. The premises of this rule are atomic assertions, and hence not the conclusion of the $\wedge i$ or $\vee i$ rules. Nor are they the conclusion of the $\wedge e$ and $\vee e$ rules, since elimination rules cannot occur in π. Thus they are the result of the ax_1 rule. This means that there are distinct nonces m, n and a term $\{b\}_k$ such that $m \prec \{b\}_k \in S_\nu$ and $n \prec \{b\}_k \in S_\nu$. This contradicts the definition of S_ν. Thus we see that the only rules occurring in π are ax_1, $\wedge i$ and $\vee i$. We now show by induction that $\nu \models \psi$ for all subproofs π' of π with conclusion $\varnothing, S_\nu \vdash \overline{\psi}$.
  - Suppose the last rule of π' is ax_1. Then $\overline{\psi} \in S_\nu$, and for some $i \leqslant n$, $\psi = p_i$ or $\psi = \neg p_i$. It can be easily seen by definition of S_ν that $\nu \models \psi$.
  - Suppose the last rule of π' is $\wedge i$. Then $\overline{\psi} = \overline{\psi_1} \wedge \overline{\psi_2}$, and $\varnothing, S_\nu \vdash_{al} \overline{\psi_1}$ and $\varnothing, S_\nu \vdash_{al} \overline{\psi_2}$. Thus, by induction hypothesis, $\nu \models \psi_1$ and $\nu \models \psi_2$. Therefore $\nu \models \psi$.
  - Suppose the last rule of π' is $\vee i$. Then $\overline{\psi} = \overline{\psi_1} \vee \overline{\psi_2}$, and either $\varnothing, S_\nu \vdash_{al} \overline{\psi_1}$ or $\varnothing, S_\nu \vdash_{al} \overline{\psi_2}$. Thus, by induction hypothesis, either $\nu \models \psi_1$ or $\nu \models \psi_2$. Therefore $\nu \models \psi$.

◀

▶ **Theorem 18.** *The derivability problem is co-NP-hard.*

4.3 Algorithm for checking derivability

Fix X_0, Φ_0 and α_0. In order to check whether $X_0, \Phi_0 \vdash \alpha_0$, we compute all assertions (from a bounded set) that can be derived from X_0 and Φ_0, and check if α_0 belongs to this set. On the face of it, it looks like more work than necessary, but recall that many assertions need to be anyway derived on the way to deriving α_0. Thus we look at all the potential assertions that may be derived in this manner.

Let $\mathsf{sf} = sf(\Phi_0 \cup \{\alpha_0\})$, $|\mathsf{sf}| = N$, and $\mathsf{st} = st(X, \Phi_0 \cup \{\alpha_0\})$. Clearly $|\mathsf{st}| <= N$. To check whether $X_0, \Phi_0 \vdash \alpha_0$, we check whether the set $derive(X_0, \Phi_0) = \{\alpha \in \mathsf{sf} \mid X_0, \Phi_0 \vdash \alpha\}$ contains α_0. Below we describe a general procedure to compute $derive(X, \Phi)$ for any $X \subseteq \mathsf{st}$ and $\Phi \subseteq \mathsf{sf}$.

For $X \subseteq st$ and $\Phi \subseteq sf$, define

$local(X, \Phi) = \{\alpha \in sf \mid X, \Phi \vdash \alpha$ has a derivation which does not use the $\vee e$ rule$\}$

A few simple observations about *derive* and *local*.

- $\Phi \subseteq local(X, \Phi) \subseteq derive(X, \Phi)$.
- $local(X, derive(X, \Phi)) = derive(X, derive(X, \Phi)) = derive(X, \Phi)$.
- If $\Phi = local(X, \Psi)$ for some Ψ, then $local(X, \Phi) = \Phi$.

▶ **Lemma 19.** *$local(X, \Phi)$ is computable in time polynomial in N.*

Proof. Let $Y = \{t \in st \mid X \vdash_{dy} t\}$. By Proposition 5, Y is computed in $O(N)$ time.

In the absence of $\vee e$, there is no branching during proof search. Hence we can compute $local(Y)$ bottom-up, as detailed below in Algorithm 1.

For $\Psi \subseteq sf$, we define $onestep(\Psi) \subseteq sf$ to be the set

$\{\alpha \in sf \mid \alpha$ is the conclusion of a rule r (other than $\vee e$) with premises $S \subseteq \Psi \cup Y\}$.

Two important observations about $onestep(\Psi)$.

- $\Psi \subseteq onestep(\Psi)$, because of the rule *ax*.
- $onestep(\Psi)$ is computable in time $O(N^2)$. This is because in all the rules other than $\vee e$, the antecedents (formulas occurring to the left of $\vdash$) in the premises are the same as the antecedents in the conclusion. Thus we need to consider only consequents (the formulas to the right of $\vdash$) in a proof. This means that we only need to consider all pairs of formulas in Ψ to compute $onestep(\Psi)$.

Since $|sf| = N$ and S increases monotonically in Algorithm 1, the *while* loop runs only for N iterations. Thus $local(\Psi)$ is computable in time $O(N^3)$. ◀

ALGORITHM 1: Algorithm to compute $local(X, \Phi)$, for $X \subseteq st$ and $\Phi \subseteq sf$

```
S ← ∅; S′ ← Φ;
while (S ≠ S′) do
    S ← S′;
    S′ ← onestep(S);
end
return S.
```

We now present the algorithm to compute $derive(X, \Phi)$. It is presented as the recursive function f in Algorithm 2.

▶ **Proposition 20** (Soundness)**.** *For $X \subseteq st$ and $\Phi \subseteq sf$, $f(X, \Phi) \subseteq derive(X, \Phi)$.*

Proof. The proof is by induction on $N - |\Phi|$. If $|\Phi| = N$, then it is easy to see that no recursive call to f would be made, and $f(X, \Phi) = local(X, \Phi) \subseteq derive(X, \Phi)$. In general, $f(X, \Phi) = local(X, \Phi \cup S)$, so it suffices to prove that $S \subseteq derive(X, \Phi)$.

So suppose $\sigma : \beta \in S$. Then there is a $\sigma : (\alpha_1 \vee \alpha_2) \in \Phi$ such that $\sigma : \alpha_1 \notin \Phi$ and $\sigma : \alpha_2 \notin \Phi$, but $\sigma : \beta \in f(X, \Phi \cup \{\sigma : \alpha_1\}) \cap f(X, \Phi \cup \{\sigma : \alpha_2\})$. But by induction hypothesis, $\sigma : \beta \in derive(X, \Phi \cup \{\sigma : \alpha_1\}) \cap derive(X, \Phi \cup \{\sigma : \alpha_2\})$. This means that there are derivations of $X, \Phi \cup \{\sigma : \alpha_1\} \vdash \sigma : \beta$ and $X, \Phi \cup \{\sigma : \alpha_2\} \vdash \sigma : \beta$. Since $\sigma : (\alpha_1 \vee \alpha_2) \in \Phi$, we also have a derivation of $X, \Phi \vdash \sigma : (\alpha_1 \vee \alpha_2)$. Combining all this using the $\vee e$ rule, we get a derivation of $X, \Phi \vdash \sigma : \beta$ and thus $\sigma : \beta \in derive(X, \Phi)$.

This proves that $S \subseteq derive(X, \Phi)$, and we are done. ◀

ALGORITHM 2: Algorithm to compute $\mathit{derive}(X, \Phi)$, for $X \subseteq st$ and $\Phi \subseteq sf$

```
function f(X, Φ)
    S ← ∅;
    for all σ : (α1 ∨ α2) ∈ Φ do
        if σ : α1 ∉ Φ and σ : α2 ∉ Φ then
            T ← f(X, Φ ∪ {σ : α1}) ;
            U ← f(X, Φ ∪ {σ : α2}) ;
            S ← S ∪ (T ∩ U) ;
        end
    end
    return local(X, Φ ∪ S)
```

▶ **Proposition 21** (Completeness)**.** *For $X \subseteq st$, $\Phi \subseteq sf$, $\mathit{derive}(X, \Phi) \subseteq f(X, \Phi)$.*

Proof. For all $\alpha \in sf$, we prove that if $\alpha \in \mathit{derive}(X, \Phi)$ then $\alpha \in f(X, \Phi)$. The proof is by induction on the structure of a derivation π of $X, \Phi \vdash \alpha$. There are two cases to consider.

- Suppose the last rule r of π is not $\vee e$. If r is ax_1 or ax_2 or eq, $\alpha \in \Phi \subseteq \mathit{local}(X, \Phi)$ and $\mathit{local}(X, \Phi) \subseteq f(X, \Phi)$. If r is something else, let $P = \{\beta \mid X, \Phi \vdash \beta \text{ is a premise of r}\}$. Since each $\beta \in P$ is the conclusion of a subproof of π, by induction hypothesis, $P \subseteq f(X, \Phi)$. Now $\alpha \in \mathit{local}(X, P) \subseteq \mathit{local}(X, f(X, \Phi)) = f(X, \Phi)$.
- Suppose the last rule of π is $\vee e$. Then α is of the form $\sigma : \beta$ (where σ could also be ε) and there are three subproofs of π with conclusions $X, \Phi \vdash \sigma : (\alpha_1 \vee \alpha_2)$ and $X, \Phi \cup \{\sigma : \alpha_1\} \vdash \sigma : \beta$, and $X, \Phi \cup \{\sigma : \alpha_2\} \vdash \sigma : \beta$, respectively. If either $\sigma : \alpha_1 \in \Phi$ or $\sigma : \alpha_2 \in \Phi$, then one of the immediate subproofs of π witnesses the fact that $X, \Phi \vdash \sigma : \beta$, and we are done, by induction hypothesis. Otherwise we see by induction hypothesis that

  $$\sigma : \beta \in f(X, \Phi \cup \{\sigma : \alpha_1\}) \cap f(X, \Phi \cup \{\sigma : \alpha_2\}).$$

  Thus $\sigma : \beta \in f(X, \Phi)$, as required.

◀

▶ **Theorem 22.** *For $X \subseteq st$ and $\Phi \subseteq sf$, $f(X, \Phi) = \mathit{derive}(X, \Phi)$.*

▶ **Theorem 23.** *The derivability problem is in PSPACE.*

Proof. The nesting depth of recursion in the function f is at most N, and the three variables S, T and U can be seen as bit vectors of size N (encoding subsets of sf). Thus we need to store at most $O(N^2)$ on the stack (and reuse space across "disjoint" invocations). All calls to the subroutine *local* can be performed using two global bit vectors, each of size N, and do not add to the space complexity. ◀

We have proved that the derivability problem is co-NP-hard, and that it is in PSPACE. This gap has been bridged in [29], where we prove a co-NP upper bound for several fragments of intuitionistic logic with disjunction. As is made clear in the proofs, the high complexity of the derivability problem is mainly due to the disjunction elimination rule, and so it is worth exploring ways to limit its effect. One simple way of achieving this is to impose an upper bound on the number of disjunctions that occur in Φ (independent of the size of Φ). The next theorem shows that this modified problem can be solved in PTIME, using the same procedure we presented earlier.

▶ **Theorem 24.** *The derivability problem with bounded number of disjunctions is solvable in PTIME.*

Proof. If the are only p disjunctions (independent of N), the height of the call tree is bounded by p, and the degree of each node in the call tree is at most N. Thus the total number of calls to f is at most N^p. Since *local* can be computed in polynomial time, this theorem follows. ◀

4.4 Safety checking

We have studied the complexity of the derivability problem, which pertains to a *passive* intruder that only derives new terms and assertions from its store of terms and assertions, without engaging with other agents actively. But the main problem of interest for the formal verification of protocols is the *active intruder deduction problem*, which asks whether is an attack on a given protocol. Formally, an attack on Pr is a run of $TS(Pr)$ that leads to an undesirable system state. Since the problem is undecidable, as mentioned earlier, we place bounds to obtain decidability. We formalize intruder deduction (the general and bounded versions) as follows.

▶ **Definition 25** (Safety checking and bounded safety checking)**.** Let Safe be an arbitrary, but fixed safety predicate (i.e. a set of protocol states).

Safety checking: Given a protocol Pr, is some protocol state $s \notin$ Safe reachable in $TS(Pr)$ from an initial protocol state?

k-*bounded safety checking*: Given Pr, is some protocol state $s \notin$ Safe with at most k-role instances reachable in $TS(Pr)$ from an initial protocol state?

▶ **Theorem 26.** 1. *If membership in Safe is decidable in PSPACE, the* k*-bounded safety checking w.r.t. Safe is solved in PSPACE.*
2. *If membership in Safe is decidable in NP, the* k*-bounded safety checking w.r.t. Safe is in NP if we restrict our attention to protocols with at most* p *disjunctions, for a fixed* p*.*

Proof. 1. A run of Pr starting from an initial state with at most k role instances is of length linear in the sum of the lengths of all roles in Pr. A PSPACE algorithm can go through all such runs to see if an unsafe protocol state is reachable. To check that each action is enabled at the appropriate protocol state along a run, we need to solve linearly many instances of the derivability problem, which runs in PSPACE. Thus the problem is in PSPACE.
2. One can guess a sequence of protocol states and actions of length linear in the size of Pr and verify that all the actions are enabled at the appropriate states. Since we are considering a protocol with at most p disjunctions for a fixed p, along each run we consider, there will be at most $k * p$ disjunctions, which is still independent of the size of the input. To check that actions are enabled at the appropriate states, we need to solve linearly many instances of the derivability problem (with bounded number of disjunctions this time) which can be done in polynomial time. Thus the problem is in NP.

◀

We leave a finer analysis of the complexity of both the derivability problem and the safety checking problem for future work.

5 Discussion

We have argued that it is worthwhile to extend the Dolev-Yao model of security protocols so that agents have the capability to communicate assertions about terms in addition to terms. These assertions play the same role as certificates that may be verified but cannot be generated by the recipient. We have suggested that such an abstraction allows us to model a variety of such certificate mechanisms. As a contribution to the theory of security protocols, we delineate the complexity of the derivability problem and provide a decision procedure. We study the safety checking problem (which involves the active intruder).

We would like to emphasize here that the main thrust of the paper is the overall framework, rather than a specific assertion language. We use a minimal logic for assertions, and many extensions by way of connectives or modalities are possible; however, it is best to drive extensions by

applications that require them. We leave the study of such extensions to future work. We have indicated another way to enrich the assertion language in Section 3. We added the ability for an assertion to contain placeholders which refer to the accompanying term. Another natural extension is to use variables in assertions to refer to various parts of the accompanying term. The formal development can be carried out on the lines of Section 3. But there are interesting considerations and technical challenges when we enrich the language with variables and other operators (like equality and inequality between terms) and connectives (like negation).

Another important aspect is the power of the intruder in these models. We should be careful that the increased power given to the honest agents of communicating assertions does not allow the intruder scope for more attacks (than in the basic Dolev-Yao model) by injecting "false" assertions into the system. Our definitions ensure that the intruder does not have this power, but it is a challenging question to determine how much we can vary the key definitions while still limiting the power of the intruder. Quite apart from the intruder's ability to inject falsehoods is the ability to learn secrets. It would be an interesting exercise to study in more detail the attacks in the presence of communicated assertions (in comparison to attacks in the basic Dolev-Yao model).

The central elements of our model are the derivation rules and the state transition rules for various actions. These define the semantics of protocols for us. We have provided an informal operational justification of the transition rules in terms of a TTP that verifies proofs. But it is desirable to provide a formal operational model that justifies the various choices made in the handling of communicated assertions. Of particular relevance here is the rely-guarantee framework for trust management proposed in [23].

APPENDIX

A Normalization and other proofs

Among the rules, ax_1, *split*, *dec* and $\wedge e$ are the *pure elimination rules*, $\vee e$ and $\perp$ are the *hybrid rules*, and the rest (ax_2, *pair*, *enc*, $\wedge i$ and $\vee i$) are the *pure introduction rules*. The premises of a pure elimination rule or a hybrid rule are classified into *major premises* and *minor premises*. For the *split*, *dec* and $\wedge e$ rules, there is only one premise of the form $X, \Phi \vdash \alpha$ and that is its major premise. For the $\vee e$ rule, the major premise is the sequent $X, \Phi \vdash \sigma : (\alpha \vee \beta)$, involving the disjunction that is eliminated. For the $\perp$ rule, both premises are major.

A *normal derivation* is one satisfying the following conditions:

- The major premise of every pure elimination rule or the $\vee e$ rule (occurring in the proof) is the conclusion of a pure elimination rule.
- The conclusion of any instance of the $\perp$ rule is not the premise of any introduction rule.
- The conclusion of any instance of the $\perp$ rule is not the major premise of any hybrid or elimination rule.

▶ **Definition 27** (Rank of a derivation)**.** Let π be a derivation with last rule r and conclusion $X, \Phi \vdash \alpha$. Let $\pi_1, \ldots, \pi_n$ be the immediate subproofs of π. Let each π_i end with rule r_i and have conclusion $X, \Phi_i \vdash \alpha_i$. Also, let $X, \Phi_1 \vdash \alpha_1$ be the major premise of r. By induction on π, we define $rank(\pi)$ as follows:

- If r is a pure elimination rule and r_1 is a hybrid rule or pure introduction rule, then

$$rank(\pi) = \max(|\alpha_1|, rank(\pi_1), \cdots, rank(\pi_n)).$$

- If r_1 is $\perp$ and r is a hybrid rule or pure elimination rule or pure introduction rule, then

$$rank(\pi) = \max(|\alpha_1|, rank(\pi_1), \cdots, rank(\pi_n)).$$

- Otherwise

$$\mathit{rank}(\pi) = \max(\mathit{rank}(\pi_1), \cdots, \mathit{rank}(\pi_n)).$$

▶ **Proposition 28** (Monotonicity). *If there is a proof of* $X, \Phi \vdash \alpha$ *with cut rank* m *and* $\Phi \subseteq \Phi'$*, then there is a proof of* $X, \Phi' \vdash \alpha$ *with cut rank* m*.*

Proof. Let π be a proof of $X, \Phi \vdash \alpha$, and let $\Phi'' = \Phi' \setminus \Phi$. It is easy to check that replacing every sequent $X, \Psi \vdash \beta$ occurring in π by $X, \Psi \cup \Phi'' \vdash \beta$, we still have a valid proof π', with conclusion $X, \Phi' \vdash \alpha$. (The point is that in rules involving a discharge of the premises, the discharge is optional, so if some rule in π discharges a formula in Φ'', we can apply the same rule in π' without discharging that formula.) Since the structure of the proof does not change, the cut rank remains the same.

◀

▶ **Proposition 29** (Admissibility of Cut). *If* π_1 *is a derivation of* $X, \Phi \vdash \alpha$ *(with last rule* r_1*) and* π *a derivation of* $X, \Psi \vdash \beta$ *(with last rule r), then there is a derivation* ϖ *of* $X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \beta$ *such that*

$$\mathit{rank}(\varpi) \leqslant \max(\mathit{rank}(\pi_1), \mathit{rank}(\pi), |\alpha|).$$

Further, either the last rule of ϖ *is r or* $\beta = \alpha$ *and the last rule of* ϖ *is* r_1*.*

Proof. The proof is by induction on the size of π, and a case analysis on r. For notational ease, we let $\mathit{rank}(\pi_1) = m_1$, $\mathit{rank}(\pi) = m$, and $n = \max(m_1, m, |\alpha|)$. We present a few sample cases below.

r is *ax*$_1$: If $\beta \neq \alpha$, then $\beta \in \Psi \setminus \{\alpha\}$ and we can take ϖ to be the following proof:

$$\frac{}{X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \beta}\, ax$$

Clearly $\mathit{rank}(\varpi) = 0 \leqslant n$ and the last rule of ϖ' is r.

If $\beta = \alpha$, then we take ϖ to be the proof of $X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \alpha$ guaranteed by Monotonicity (applied to π_1). Clearly $\mathit{rank}(\varpi) = m_1 \leqslant n$.

r is $\wedge$i: Then π has the following structure:

$$\frac{\begin{array}{cc}\tau_1 & \tau_2\\ \vdots & \vdots\\ X, \Psi \vdash \beta_1 & X, \Psi \vdash \beta_2\end{array}}{X, \Psi \vdash \beta}\, \wedge i$$

By induction hypothesis, there exist proofs ϖ_1 and ϖ_2 with conclusions respectively $X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \beta_1$ and $X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \beta_2$, both of which have cut ranks at most n. We define ϖ to be the following proof:

$$\frac{\begin{array}{cc}\varpi_1 & \varpi_2\\ \vdots & \vdots\\ X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \beta_1 & X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \beta_2\end{array}}{X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \beta}\, \wedge i$$

Clearly $\mathit{rank}(\varpi) = \max(\mathit{rank}(\varpi_1), \mathit{rank}(\varpi_2)) \leqslant n$. Further, the last rule of ϖ is r.

r is $\vee e$: Then π has the following structure:

$$\frac{\begin{array}{ccc}\tau_1 & & \\ \vdots & & \\ \hline X, \Psi \vdash \sigma : (\varphi \vee \psi) & r_1' & \end{array} \quad \begin{array}{c}\tau_2\\ \vdots\\ X, \Psi \cup \{\sigma : \varphi\} \vdash \beta\end{array} \quad \begin{array}{c}\tau_3\\ \vdots\\ X, \Psi \cup \{\sigma : \psi\} \vdash \beta\end{array}}{X, \Psi \vdash \beta}\, \vee e$$

By induction hypothesis, there exist proofs ϖ_1, ϖ_2 and ϖ_3 with conclusions respectively $X, \Phi \cup (\Psi \setminus \{\alpha\}) \vdash \sigma : (\varphi \vee \psi)$, $X, \Phi \cup ((\Psi \cup \{\sigma : \varphi\}) \setminus \{\alpha\}) \vdash \beta$ and $X, \Phi \cup ((\Psi \cup \{\sigma : \psi\}) \setminus \{\alpha\}) \vdash \beta$, all of whose cut ranks are $\leqslant n$. By appealing to Monotonicity if necessary (in the cases when α is $\sigma : \varphi$ or $\sigma : \psi$), we can take the conclusion of ϖ_2 and ϖ_3 to be $X, \Phi \cup \{\sigma : \varphi\} \cup (\Psi \setminus \{\alpha\}) \vdash \beta$ and $X, \Phi \cup \{\sigma : \psi\} \cup (\Psi \setminus \{\alpha\}) \vdash \beta$. ϖ is the following proof (using Θ to denote $\Phi \cup (\Psi \setminus \{\alpha\})$):

$$\dfrac{\dfrac{\begin{matrix}\varpi_1 \\ \vdots \\ X, \Theta \vdash \sigma : (\varphi \vee \psi)\end{matrix}}{}\, r_1'' \qquad \begin{matrix}\varpi_2 \\ \vdots \\ X, \Theta \cup \{\sigma : \varphi\} \vdash \beta\end{matrix} \qquad \begin{matrix}\varpi_3 \\ \vdots \\ X, \Theta \cup \{\sigma : \psi\} \vdash \beta\end{matrix}}{X, \Theta \vdash \beta}\, \vee e$$

Now if r_1'' is a pure elimination, $\mathit{rank}(\varpi) \leqslant n$. Otherwise, $\mathit{rank}(\varpi) \leqslant \max(|\varphi \vee \psi|, n)$. But then either $r_1'' = r_1'$ (in which case $|\sigma : (\varphi \vee \psi)| \leqslant m \leqslant n$), or $\alpha = \sigma : (\varphi \vee \psi)$ and $r_1'' = r_1$ (in which case $|\sigma : (\varphi \vee \psi)| = |\alpha| \leqslant n$). Thus $\mathit{rank}(\varpi) \leqslant n$. Again the last rule of ϖ is r.

◀

▶ **Lemma 30.** *Suppose π is a derivation with conclusion $X, \Phi \vdash \alpha$ and last rule r with $\mathit{rank}(\pi) = m > 0$, and all proper subderivations of π are of rank $< m$. Then the following hold.*

1. *If r is a pure elimination rule, $|\alpha| < m$.*
2. *There is a derivation π' of $X, \Phi \vdash \alpha$ such that $\mathit{rank}(\pi') < m$.*

Proof. Let $\pi_1, \ldots, \pi_n$ be the immediate subproofs of π. Let each π_i end with rule r_i and have conclusion $X, \Phi_i \vdash \alpha_i$, and let $X, \Phi_1 \vdash \alpha_1$ be the major premise of r. Given the conditions of the lemma, it is clear that $\mathit{rank}(\pi) = |\alpha_1| = m$ and one of the following cases holds:

- r_1 is not a pure elimination rule, r is a pure elimination rule or a hybrid rule, and $X_1 = X$.
- r_1 is $\bot$ and r is a pure introduction rule.

1. If r is a pure elimination rule, then $\alpha_1 = \alpha \wedge \beta$ or $\alpha_1 = \beta \wedge \alpha$, for some β. But then it is clear that $|\alpha| < |\alpha_1| = m$.
2. To show the existence of π', we perform an induction on $||\pi||$ and a case analysis on r_1. A few representative cases are shown below.

   - It cannot be the case that r_1 is *ax*$_2$ or *eq*, since the conclusions of these rules are atomic formulas, and there can be no hybrid or pure elimination rules with them as major premises.
   - Suppose r_1 is *pair*. Then r has to be *split*. In this case we can take π' to be one of the immediate subproofs of π_1, and clearly $\mathit{rank}(\pi') < m$.
   - Suppose r_1 is *enc*. Then r has to be *dec*. In this case we can take π' to be one of the immediate subproofs of π_1, and clearly $\mathit{rank}(\pi') < m$.
   - Suppose r_1 is $\wedge i$. Then r has to be $\wedge e$. In this case we can take π' to be one of the immediate subproofs of π_1, and clearly $\mathit{rank}(\pi') < m$.
   - Suppose r_1 is $\vee i$. Then r has to be $\vee e$. Say $\alpha_1 = \beta \vee \gamma$ and the major premise of r_1 is β. Note that $|\beta| < |\beta \vee \gamma| = m$. Let π_2 be the immediate subproof of π with conclusion $X, \Phi \cup \{\beta\} \vdash \alpha$, and let π_{11} be the subproof of π_1 with conclusion $X, \Phi \vdash \beta$. Thus we can apply cut on π_{11} and π_2 to get a derivation π' of $X \vdash \alpha$ such that

     $$\mathit{rank}(\pi') \leqslant \max(|\beta|, \mathit{rank}(\pi_{11}), \mathit{rank}(\pi_2)) < m.$$

- Suppose r_1 is $\vee e$. Now r can be any pure elimination or hybrid rule. We consider the case when it is $\vee e$. The rest of the cases are similar. Now $\alpha_1 = \beta \vee \beta'$ and π has the following form:

$$\dfrac{\dfrac{\begin{matrix}\pi_{11}\\ \vdots\\ X,\Phi\vdash\gamma\vee\gamma'\end{matrix}\quad\begin{matrix}\pi_{12}\\ \vdots\\ X,\Phi\cup\{\gamma\}\vdash\beta\vee\beta'\end{matrix}\quad\begin{matrix}\pi_{13}\\ \vdots\\ X,\Phi\cup\{\gamma'\}\vdash\beta\vee\beta'\end{matrix}}{X,\Phi\vdash\beta\vee\beta'}\vee e\quad\begin{matrix}\pi_2\\ \vdots\\ X,\Phi\cup\{\beta\}\vdash\alpha\end}\quad\begin{matrix}\pi_3\\ \vdots\\ X,\Phi\cup\{\beta'\}\vdash\alpha\end}{X,\Phi\vdash\alpha}\vee e$$

Let τ_2 be the following proof

$$\dfrac{\begin{matrix}\pi_{12}\\ \vdots\\ X,\Phi\cup\{\gamma\}\vdash\beta\vee\beta'\end{matrix}\quad\begin{matrix}\pi_2\\ \vdots\\ X,\Phi\cup\{\gamma,\beta\}\vdash\alpha\end}\quad\begin{matrix}\pi_3\\ \vdots\\ X,\Phi\cup\{\gamma,\beta'\}\vdash\alpha\end}{X,\Phi\cup\{\gamma\}\vdash\alpha}\vee e$$

and let τ_3 be the following proof.

$$\dfrac{\begin{matrix}\pi_{13}\\ \vdots\\ X,\Phi\cup\{\gamma'\}\vdash\beta\vee\beta'\end{matrix}\quad\begin{matrix}\pi_2\\ \vdots\\ X,\Phi\cup\{\gamma',\beta\}\vdash\alpha\end}\quad\begin{matrix}\pi_3\\ \vdots\\ X,\Phi\cup\{\gamma',\beta'\}\vdash\alpha\end}{X,\Phi\cup\{\gamma'\}\vdash\alpha}\vee e$$

Now it is possible that $\mathit{rank}(\tau_2) = \mathit{rank}(\tau_3) = m$, but $\|\tau_2\| < \|\pi\|$ and $\|\tau_3\| < \|\pi\|$. Hence by induction hypothesis, there are proofs π_2' and π_3', both of cut rank $< m$, with conclusions $X,\Phi\cup\{\gamma\}\vdash\alpha$ and $X,\Phi\cup\{\gamma'\}\vdash\alpha$ respectively. We take π' to be the following proof:

$$\dfrac{\begin{matrix}\pi_{11}\\ \vdots\\ X,\Phi\vdash\gamma\vee\gamma'\end{matrix}\quad\begin{matrix}\pi_2'\\ \vdots\\ X,\Phi\cup\{\gamma\}\vdash\alpha\end}\quad\begin{matrix}\pi_3'\\ \vdots\\ X,\Phi\cup\{\gamma'\}\vdash\alpha\end}{X,\Phi\vdash\alpha}\vee e$$

Now if π_{11} ends in a pure elimination,

$$\mathit{rank}(\pi') = \max(\mathit{rank}(\pi_{11}), \mathit{rank}(\pi_2'), \mathit{rank}(\pi_3')) < m.$$

Otherwise $\mathit{rank}(\pi') \leqslant \max(m-1, |\gamma\vee\gamma'|)$. But if π_{11} does not end in a pure elimination, $|\gamma\vee\gamma'| \leqslant \mathit{rank}(\pi_1) < m$, and it follows that $\mathit{rank}(\pi') < m$.

- Suppose r_1 is $\bot$. Now r can be any rule. We consider the case when it is $\vee e$. The rest of the cases are similar. Now $\alpha_1 = \beta \vee \beta'$ and π has the following form:

$$\dfrac{\dfrac{\begin{matrix}\pi_{11}\\ \vdots\\ X,\Phi\vdash m\prec\{b\}_k\end}\quad\begin{matrix}\pi_{12}\\ \vdots\\ X,\Phi\vdash m\prec\{b\}_k\end}{X,\Phi\vdash\beta\vee\beta'}\bot(m\neq n)\quad\begin{matrix}\pi_2\\ \vdots\\ X,\Phi\cup\{\beta\}\vdash\alpha\end}\quad\begin{matrix}\pi_3\\ \vdots\\ X,\Phi\cup\{\beta'\}\vdash\alpha\end}{X,\Phi\vdash\alpha}\vee e$$

Let τ be the following proof

$$\dfrac{\begin{matrix}\pi_{11}\\ \vdots\\ X,\Phi\vdash m\prec\{b\}_k\end}\quad\begin{matrix}\pi_{12}\\ \vdots\\ X,\Phi\vdash m\prec\{b\}_k\end}{X,\Phi\vdash\alpha}\bot(m\neq n)$$

Now it is possible that $\mathit{rank}(\tau) = m$, but $\|\tau\| < \|\pi\|$. Hence by induction hypothesis, there is a proof π' of cut rank $< m$ with conclusion $X,\Phi\vdash\alpha$.

◀

▶ **Theorem 31** (Weak normalization). *If there is a derivation of* $X, \Phi \vdash \alpha$ *then there is a normal derivation of* $X, \Phi \vdash \alpha$.

Proof. For every derivation π, define $\mu(\pi)$ to be the pair (d, n) where $d = \mathit{rank}(\pi)$, and n is the number of subderivations of π of rank d. If $\mathit{rank}(\pi) = 0$, π is already normal. If not, let $\mathit{rank}(\pi) = d > 0$ and let ϖ be a subderivation of π with conclusion $X, \Psi \vdash \beta$ such that $\mathit{rank}(\varpi) = d$ and no proper subderivation of ϖ is of rank $\geqslant d$. By Lemma 30, $|\beta| < d$ and there is another derivation ϖ' with $\mathit{rank}(\varpi') < d$ and whose conclusion is $X, \Psi \vdash \beta$. Replace ϖ by ϖ' in π to get the proof π'. Now one subderivation of rank d has been eliminated in the process of going from π to π'. But we need to check that no new derivations of rank $\geqslant d$ have been introduced in π'. The only way this can happen is if ϖ' ends in an unsafe rule and is the major premise of an elimination rule in π'. But then either $|\beta| < d$, or ϖ' ends in $\vee e$. In either case, no new subderivation of rank $\geqslant d$ gets introduced. Thus $\mu(\pi') < \mu(\pi)$. Since lexicographic ordering on pairs of natural numbers is a well order, by repeating the above process we eventually reach a proof of rank 0 – a normal proof, in other words.

◀

▶ **Definition 32.** The set of subformulas of α, denoted $\mathit{sf}(\alpha)$, is defined as the smallest set $\Phi \subseteq \mathscr{A}$ such that

- $\alpha \in \Phi$
- If $\sigma : (\beta \wedge \gamma) \in \Phi$ or $\sigma : (\beta \vee \gamma) \in \Phi$, then $\{\sigma : \beta, \sigma : \gamma\} \subseteq \Phi$.
- If $m \prec \{t\}_k \in \Phi$ or $m \prec (t, t') \in \Phi$ or $m \prec (t', t) \in \Phi$, then $m \prec t \in \Phi$.

The set of terms occurring in Φ, denoted $\mathit{terms}(\Phi)$, is defined to be

$$\{m, t \mid \sigma : (m \prec t) \in \mathit{sf}(\Phi)\} \cup \{t, t' \mid \sigma : (t = t') \in \mathit{sf}(\Phi)\}.$$

$\mathit{st}(X, \Phi)$ is defined to be $\mathit{st}(X \cup \mathit{terms}(\Phi))$ and $\mathit{sf}(X, \Phi)$ is defined to be

$$\mathit{sf}(\Phi) \cup \{m \prec \{b\}_k | m, b, k \in \mathit{st}(X, \Phi) \cap \mathscr{B}\}.$$

It is clear that $|\mathit{sf}(X, \Phi)| < 2.N^3$, where N is the size of X, Φ.

▶ **Theorem 33** (Subformula property). *Let* π *be a normal derivation with conclusion* $X, \Phi \vdash \alpha$ *and last rule* r. *Let* $X, \Phi' \vdash \beta$ *and* $X \vdash_{dy} u$ *occur in* π. *Then* $\Phi' \subseteq \mathit{sf}(\Phi)$, $\beta \in \mathit{sf}(X, \Phi \cup \{\alpha\})$ *and* $u \in \mathit{st}(X, \Phi \cup \{\alpha\})$. *Furthermore, if* r *is a pure elimination rule, then* $\beta \in \mathit{sf}(X, \Phi)$ *and* $u \in \mathit{st}(X, \Phi)$.

Proof. The proof is by induction on the structure of π, and based on a case analysis on r. We present a few representative cases here.

- Suppose r is ax_1. Then $\alpha \in \Phi$ and π is of the form

$$\frac{}{X, \Phi \vdash \alpha}\, ax_1$$

  Clearly $\beta = \alpha$ and $\Phi' = \Phi$. It follows that $\beta \in \mathit{sf}(X, \Phi)$ and $\Phi' \subseteq \mathit{sf}(\Phi)$.
- Suppose r is ax_2. Then π is of the form

$$\frac{X \vdash_{dy} m}{X, \Phi \vdash m \prec m}\, ax_2$$

  Clearly $\beta = \alpha \in \mathit{sf}(X, \Phi \cup \{\alpha\})$ and $\Phi' = \Phi \subseteq \mathit{sf}(\Phi)$. If $X \vdash_{dy} u$ occurs in π, then $u = m \in \mathit{st}(X, \Phi \cup \{\alpha\})$.

- Suppose r is *pair*. Then $\alpha = m \prec (t, t')$ and π is of the following form:

$$\dfrac{\begin{array}{ccc}\pi' & & \\ \vdots & & \\ X \vdash_{dy} (t, t') & X, \Phi \vdash m \prec t & X \vdash_{dy} st(t') \cap \mathscr{B}\end{array}}{X, \Phi \vdash m \prec (t, t')}\ pair$$

  Clearly $m \prec t \in sf(X, \alpha)$. Now either $\Phi' = \Phi$ and $\beta = \alpha$ or $X, \Phi' \vdash \beta$ occurs in π'. In the second case, by induction hypothesis $\Phi' \subseteq sf(\Phi)$ and $\beta \in sf(X, \Phi \cup \{m \prec t\})$. But $sf(X, \Phi \cup \{m \prec t\}) \subseteq sf(X, \Phi \cup \{m \prec (t, t')\})$, and hence we are done. If $X \vdash_{dy} u$ occurs in π, $u = (t, t')$ or $u \in st(t')\mathrm{cap}\mathscr{B}$ or $u \in st(X, \Phi \cup \{m \prec t\})$ (by induction hypothesis), and hence $u \in st(X, \Phi \cup \{m \prec (t, t')\})$.
- Suppose r is *enc*. The proof is similar to the previous case.
- Suppose r is *split*. Then $\alpha = m \prec t$ and π is of the following form:

$$\dfrac{\begin{array}{ccc}\pi' & & \\ \vdots & & \\ X, \Phi \vdash m \prec (t, t') & X \vdash_{dy} st(t') \cap \mathscr{B} & m \notin st(t')\end{array}}{X, \Phi \vdash m \prec t}\ split$$

  Since π is a normal proof, the last rule of π' is a pure elimination. Hence, by induction hypothesis, $m \prec (t, t') \in sf(X, \Phi)$, and hence $m \prec t \in sf(X, \Phi)$ as well. It follows by induction hypothesis that for any $X, \Phi' \vdash \beta$ occurring in π, $\beta \in sf(X, \Phi)$ and $\Phi' \subseteq sf(\Phi)$. If $X \vdash_{dy} u$ occurs in π, $u \in st(t') \cap \mathscr{B}$ or $u \in st(X, \Phi)$ (by induction hypothesis), and hence $u \in st(X, \Phi \cup \{m \prec (t, t')\}) = st(X, \Phi)$.
- Suppose r is *dec*. The proof is similar to the previous case.
- Suppose r is $\wedge i$. Then $\alpha = \sigma : (\alpha' \wedge \alpha'')$ and π is of the following form:

$$\dfrac{\begin{array}{cc}\pi' & \pi'' \\ \vdots & \vdots \\ X, \Phi \vdash \sigma : \alpha' & X, \Phi \vdash \sigma : \alpha''\end{array}}{X, \Phi \vdash \alpha}\ \wedge i$$

  Clearly $\sigma : \alpha' \in sf(X, \alpha)$ and $\sigma : \alpha'' \in sf(X, \alpha)$. Now either $\Phi' = \Phi$ and $\beta = \alpha$ or $X, \Phi' \vdash \beta$ occurs in π' or π''. In the second and third cases, by induction hypothesis $\Phi' \subseteq sf(\Phi)$ and $\beta \in sf(X, \Phi \cup \{\sigma : \alpha', \sigma : \alpha''\})$. But then $\beta \in sf(X \cup \{\alpha\})$, since $sf(X, \Phi \cup \{\sigma : \alpha', \sigma : \alpha''\}) \subseteq sf(X, \Phi \cup \{\alpha\})$. If $X \vdash_{dy} u$ occurs in π, then by induction hypothesis $u \in st(X, \Phi \cup \{\sigma : \alpha', \sigma : \alpha''\}) \subseteq st(X, \Phi \cup \{\alpha\})$.
- Suppose r is $\vee i$. The proof is similar to the above case.
- Suppose r is $\wedge e$. The proof is similar to the *split* case.
- Suppose r is $\vee e$. Then π is of the following form:

$$\dfrac{\begin{array}{ccc}\pi_1 & \pi_2 & \pi_3 \\ \vdots & \vdots & \vdots \\ X, \Phi \vdash \sigma : (\varphi \vee \psi) & X, \Phi \cup \{\sigma : \varphi\} \vdash \alpha & X, \Phi \cup \{\sigma : \psi\} \vdash \alpha\end{array}}{X, \Phi \vdash \alpha}\ \vee e$$

  Again, either $\Phi' = \Phi$ and $\beta = \alpha$ or $X, \Phi' \vdash \beta$ occurs in one of the π_i's. Suppose it occurs in π_1. Notice that since π is normal, the last rule of π_1 is a pure elimination, and hence by induction hypothesis, $\Phi' \subseteq sf(\Phi)$ and $\beta \in sf(X, \Phi)$. In particular, since $\sigma : (\varphi \vee \psi)$ occurs in π_1, $\sigma : (\varphi \vee \psi) \in sf(X, \Phi)$ and thus $\{\sigma : \varphi, \sigma : \psi\} \subseteq sf(X, \Phi)$. Now suppose that $X, \Phi' \vdash \beta$ occurs in π_2 or π_3. Then by induction hypothesis, we have that $\Phi' \subseteq sf(\Phi \cup \{\sigma : \varphi, \sigma : \psi\}) \subseteq sf(\Phi)$, and $\beta \in sf(X, \Phi \cup \{\sigma : \varphi, \sigma : \psi, \alpha\}) \subseteq sf(X, \Phi \cup \{\alpha\})$.

- Suppose r is $\bot$. Then π is of the following form:

$$\dfrac{\begin{array}{cc}\pi_1 & \pi_2\\ \vdots & \vdots\\ X, \Phi \vdash \sigma : (m \prec \{b\}_k) & X, \Phi \vdash \sigma : (n \prec \{b\}_k)\end{array}}{X, \Phi \vdash \sigma : \alpha}\ \bot\ (m \neq n)$$

Suppose π_1 is of the following form

$$\dfrac{\begin{array}{c}\pi_{11}\\ \vdots\ r'\\ X, \Phi \vdash m \prec b \qquad \cdots\end{array}}{X, \Phi \vdash m \prec \{b\}_k}\ enc$$

Now r' is a pure elimination rule or ax_2. In the first case, we have $m, b \in st(X, \Phi)$ and $k \in st(X)$, and hence $m \prec \{b\}_k \in sf(X, \Phi)$. In the second case $m, b, k \in st(X)$ and hence $m \prec \{b\}_k \in sf(X, \Phi)$. Otherwise π ends in a pure elimination rule, and by induction hypothesis, $\sigma : (m \prec \{b\}_k) \in sf(X, \Phi)$. Similarly $\sigma : (n \prec \{b\}_k) \in sf(X, \Phi)$ as well.
Now $\beta = \alpha$ and $\Phi' = \Phi$ or $X, \Phi' \vdash \beta$ occurs in one of the subproofs. We conclude by induction hypothesis that $\Phi' \subseteq sf(\Phi)$ and

$$\beta \in sf(X, \Phi \cup \{\alpha, \sigma : (m \prec \{b\}_k), \sigma : (n \prec \{b\}_k)) \subseteq sf(X, \Phi \cup \{\alpha\}).$$

◀

References

1 Martin Abadi and Roger M. Needham. Prudent engineering practices for cryptographic protocols. *IEEE Trans. Software Engineering*, 22:6–15, 1996.

2 Ross Anderson and Roger M. Needham. Programming satan's computer. In *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 426–441. Springer, 1995.

3 Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Type-checking zero-knowledge. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008*, pages 357–370, 2008.

4 Michael Backes, Matteo Maffei, and Dominique Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *2008 IEEE Symposium on Security and Privacy (S&P 2008)*, pages 202–215, 2008.

5 A. Baskar, Prasad Naldurg, K. R. Raghavendra, and S. P. Suresh. Primal Infon Logic: Derivability in Polynomial Time. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 163–174, 2013.

6 A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '07, pages 62–71, 2007.

7 A. Baskar, R. Ramanujam, and S.P. Suresh. A dolev-yao model for zero knowledge. In *Advances in Computer Science - ASIAN 2009. Information Security and Privacy*, volume 5913 of *Lecture Notes in Computer Science*, pages 137–146. 2009.

8 A. Baskar, R. Ramanujam, and S.P. Suresh. A dexptime-complete dolev-yao theory with distributive encryption. In *Mathematical Foundations of Computer Science 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 102–113. 2010.

9 Josh Cohen Benaloh. *Advances in Cryptology — CRYPTO' 86: Proceedings*, chapter Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols, pages 213–222. 1987.

10 Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, JUN 2001.

11 Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, February 1990.

12 Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. *Theoretical Computer Science*, 338(1-3):247–274, 2005.

13 Hubert Comon-Lundh, Véronique Cortier, and Eugen Zălinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Trans. Comput. Logic*, 11(2):9:1–9:42.

14 Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003),*, pages 271–282, 2003.

15 Cas J. F. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008*, pages 414–418, 2008.

16 Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

17 Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

18 Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

19 Georg Fuchsbauer and David Pointcheval. Anonymous consecutive delegation of signing rights: Unifying group and proxy signatures. *IACR Cryptology ePrint Archive*, 2008:37, 2008.

20 Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques,*, pages 244–251, 1992.

21 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

22 Yuri Gurevich and Itay Neeman. Logic of infons: The propositional case. *ACM Trans. Comput. Logic*, 12(2):9:1–9:28, January 2011.

23 Joshua D. Guttman, F. Javier Thayer, Jay A. Carlson, Jonathan C. Herzog, John D. Ramsdell, and Brian T. Sniffen. Trust management in strand spaces: A rely-guarantee method. In *Programming Languages and Systems, 13th European Symposium on Programming, ESOP 2004,*, pages 325–339, 2004.

24 Nevin Heintze and J. D. Tygar. A model for secure protocols and their compositions. *IEEE Trans. Software Eng.*, 22(1):16–30, 1996.

25 Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005*, pages 186–200, 2005.

26 Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for the equational theory of abelian groups with distributive encryption. *Inf. Comput.*, 205(4):581–623, 2007.

27 Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

28 R. Ramanujam, Vaishnavi Sundararajan, and S. P. Suresh. Extending dolev-yao with assertions. In *Information Systems Security - 10th International Conference, ICISS 2014*, pages 50–68, 2014.

29 R. Ramanujam, Vaishnavi Sundararajan, and S. P. Suresh. The complexity of disjunction in intuitionistic logic. In *Logical Foundations of Computer Science - International Symposium, LFCS 2016*, pages 349–363, 2016.

30 R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

31 R. Ramanujam and S. P. Suresh. A (restricted) quantifier elimination for security protocols. *Theor. Comput. Sci.*, 367(1):228–256, November 2006.

32 Zuzana Rjaskova. Electronic voting schemes. Master's thesis, Comenius University, 2002.

33 Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions, composed keys is np-complete. *Theor. Comput. Sci.*, 1-3(299):451–475, 2003.