Extending Dolev-Yao with assertions[⋆]

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

[1] 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 require principals to send certifications asserting partial knowledge of terms (for instance, that an encrypted secret is 0 or 1). Such certificates are themselves modelled by cryptographic primitives or sequences of communications. For logical analysis of such protocols based on the Dolev-Yao model [12], we suggest that it is useful to separate terms and assertions about them in communications. We propose a perfect assertion assumption by which the underlying model ensures the correctness of the assertion when it is generated. The recipient may then rely on the certificate but may only forward it as second-hand information. We use a simple propositional modal assertion language involving disjunction (for partial knowledge) and formulas of the form *A says* α (for delegation). We study the complexity of the term derivability problem and *safety checking* in the presence of an active intruder (for bounded protocols). We show that assertions add complexity to verification, but when they involve only boundedly many disjunctions, the complexity is the same as that of the standard Dolev-Yao model.

1 Motivation

1.1 Assertions as certification

Formal verification of cryptographic protocols requires an abstract model of agents' capabilities and communications, and the **Dolev-Yao model** [12] has been the bulwark of such modelling. Its central elements are a *message abstraction* that views the message space as a term algebra and *term derivation rules* that specify how an agent derives new terms from old.

This model and its various extensions have been the object of study for the last 30 years. Typical extensions cater to more complex cryptographic operations like homomorphic encryption, blind signatures etc., that are useful in applications like e-voting and contract signing [15, 6, 11]. The interaction between the operators can have significant impact on term derivability, and forms an important line of theoretical research [9, 15, 10].

An important feature of the Dolev-Yao model is that it treats terms as tokens that can be copied and passed along. A recipient of a term "owns" it and can send it to others

⋆ We thank A. Baskar for discussions and comments on many ideas in the paper. We also thank the reviewers for many suggestions that improved the paper.

in its own name. On the other hand, cryptographic protocols often use *certificates* that can be verified but not "owned", i.e., an agent B which receives a certificate from A cannot later send the same certificate to C in its own name. Zero-knowledge protocols, for example, often involve agents certifying (via zero-knowledge proofs) that the terms they send to other agents possess certain properties, without revealing the entire terms. The recipient, if malicious, might try to forward this term to some other agent in its own name – however, the fact that the agent is required to have access to the entire term in order to construct the requisite certificate disallows it from being forwarded. Here are a few scenarios that illustrate the use of such certification, or *assertions*.

– A server S generates a session key k for A and B, and also certifies that the key is "good" for use by them.
– An agent A sends a vote v (encrypted) to B, along with a certificate that the vote is valid, i.e. v takes on only a few (pre-specified) values. It is not 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 [17] for more examples of this kind.)
– A passes on to B an assertion α made by S about a key k, but stating explicitly that S *says* α. (Assertions of this kind are used in [7].)

Such certification is expressed in the Dolev-Yao model in a variety of ways.

– In some cases, one relies on conventions and features of the framework, and does not explicitly model assertions in the protocol at all. Examples would be assertions about the goodness of keys, freshness of nonces etc.
– In other cases, one uses cryptographic devices like zero knowledge proofs, bit-commitment schemes etc. to make partial secrecy assertions. For example, a voter V might post an encrypted vote v 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 knowing the value of the vote.
– Sometimes one uses ad hoc conventions specific to a protocol to model some assertions. For instance, a term that models an assertion α might be paired (or tagged) with the agent name S to signify S *says* α.

In this paper, we propose an extension to the Dolev-Yao model in which agents have the ability to communicate assertions explicitly, rather than via an encoding. The fact that the above examples can be expressed in the Dolev-Yao model via various methods of translation seems to suggest that this ability does not add any expressive power. However, communicating both data and assertions 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 logical language of assertions should be seen in this light.

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]. In which case, should the proof thus sent be verified by the recipient?

While such an approach is adopted in models that view assertions as terms, in this work, the syntactic separation of terms and assertions allows us an abstraction similar to the perfect encryption assumption in the Dolev-Yao model. We call it the *perfect assertion assumption*: the model ensures the correctness of assertions at the point of generation, and honest principals assume such correctness and proceed. Thus the onus of verification shifts from the receiver of the assertion to the underlying system. This can be realized as a trusted third party (TTP) model, which verifies any proof that is sent out by any agent into the system. Note that since the adversary monitors the network, it can also see the proof sent by any agent to the TTP and replay assertions in agents' names, and therefore, adversary capabilities are not restricted much. Also note that the TTP is not a principal but an operational realisation of the network's capability to verify every proof that is sent out by agents. Thus we model assertions as signed by the sender, but not encrypted with any key (including that of the TTP). This will be explained further after we define the operational semantics of protocols, in Section 2.4.

How are assertions in our model contrasted with those with zero-knowledge primitives as in [3] or with those in which terms encode zero-knowledge proofs? Consider a simple example of an encrypted vote $\{v\}_k$, where v is either 0 or 1, and the recipient does not know k (or its inverse). In order to assert this using terms, one would present a one-out-of-2-encryption proof for v belonging to $\{0, 1\}$, as presented in [17]. In the model in [3], this would be coded up as $\mathsf{zk}_{2,1,\beta_1=\mathsf{enc}(\alpha_1,\alpha_2)\wedge(\alpha_1=0\vee\alpha_1=1)}(v, k; \{v\}_k)$. In our model, it is simply represented by the assertion $\{0 \prec \{v\}_k \vee 1 \prec \{v\}_k\}$.

1.2 Assertions in protocols, and possible attacks

We now present a few example protocols illustrating how our model can be used. We also present some attacks where the malicious intruder targets the fact that assertions are transmitted, and tries to gain more information than she is entitled to. This motivates the need for studying the verification problem for protocols with assertions.

Example 1. An agent A sends two encrypted votes to a tallier T, one choice for each of two 'positions', and accompanies each vote with a disjunctive assertion about it. The disjunction may be thought of as expressing that the vote is for one of two possible candidates interested in the position. The tallier checks these assertions, and on confirming that they are indeed true, sends a confirmation c_A to A. Otherwise it sends a 0.

– $A \rightarrow T : \{v_1\}_{k_A},\ \ \{(a\ \textit{occurs in}\ \{v_1\}_k) \vee (b\ \textit{occurs in}\ \{v_1\}_k)\}$
– $A \rightarrow T : \{v_2\}_{k_A},\ \ \{(c\ \textit{occurs in}\ \{v_2\}_k) \vee (d\ \textit{occurs in}\ \{v_2\}_k)\}$
– If T confirms these two assertions, $T \rightarrow A : c_A$. Otherwise, $T \rightarrow A : 0$.

Now consider the situation where a particular candidate, say candidate a, is interested in both positions, i.e., $c = a$ in the above protocol. Now, if agent A votes for candidate a (i.e. $v_1 = v_2 = a$) for both positions, he/she sends the same term $\{a\}_{k_A}$ twice, with two disjunctive assertions about it. The intruder can now perform disjunction elimination on these assertions to know the fact that A's vote is for candidate a.

Example 2. An agent B sends an assertion as a response to a communication by another agent A.

– $A \to B : \{m\}_{pk(B)}$
– $B \to C : \{m\}_{pk(B)}, \Big\{(a \textit{ occurs in } \{m\}_{pk(B)}) \vee (b \textit{ occurs in } \{m\}_{pk(B)})\Big\}$

A sends B a term which is an encrypted nonce, about which B generates an assertion, and then passes on both the term and the assertion to agent C. B, while generating this disjunction, takes the actual assertion (say a *occurs in* $\{m\}_{pk(B)}$), and adds a disjunct of the form r *occurs in* $\{m\}_{pk(B)}$, where $r \neq a$ is chosen at random from the constants of the protocol. This allows the intruder an attack, whereby she begins a new session with B by replaying the same term $\{m\}_{pk(B)}$ from the session of A with B. Now B sends to C the same term with a new assertion $\Big\{(a \textit{ occurs in } \{m\}_{pk(B)}) \vee (r' \textit{ occurs in } \{m\}_{pk(B)})\Big\}$, where r' is another random constant of the protocol. In the earlier session, B sent C $\Big\{(a \textit{ occurs in } \{m\}_{pk(B)}) \vee (r \textit{ occurs in } \{m\}_{pk(B)})\Big\}$. By disjunction elimination, the intruder (and C) can infer that a occurs in $\{m\}_{pk(B)}$.

Example 3. Here is a scenario that might occur in contract signing protocols. Agents A and S are interested in buying and selling an object, respectively. A commits to a value a, by sending S a term $\{v_a\}_{k(A,S)}$ and an accompanying assertion of the form a *occurs in* $\{v_a\}_{k(A,S)}$, where a is his bid for the object. S, however, is not interested in honouring A's commitment, and, without responding to A, sends agent B the assertion A *says* $\{a \textit{ occurs in } \{v_a\}_{k(A,S)}\}$. B, who is interested in buying this object at any cost, now quotes a price higher than what A quotes, and the seller S thereby gets an unfair advantage.

1.3 Logicization and challenges

The above examples motivate the formal study of Dolev-Yao with assertions. But there are several questions that need to be addressed in building a formal model, and several consequences of the choices we make. We discuss some of them here.

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* [14, 5] (which reasons about access control). A priori, certification in cryptographic protocols reveals partial information about hidden (encrypted) terms, and hence we need assertions that achieve this. We use atomic assertions about term structure and disjunctions to make the revelations partial. For instance, $(0 \textit{ occurs in } t) \vee (1 \textit{ occurs in } t)$ 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 may exclude the assertion 1 *occurs in* $\{m\}_k$. With conjunctions to bundle assertions together, we have a restricted propositional language.

The modality we study is one that refers to agents passing assertions along, and has the flavour of *delegation*: A sending α to B allows B to send A *says* α to other agents, without requiring B to be able to derive α. Many papers which view assertions as terms work with assertions similar to the ones used here. For instance, [8] presents a new cryptographic primitive for partial information transmission, while [13] deals with delegation and signatures, although there the focus is more on anonymity and group signatures.

We conduct a proof theoretic investigation of passive intruder capabilities, and we also illustrate the use of these assertions for exploring active intruder attacks. The passive intruder deduction problem (or the term derivability problem) is co-NP-hard even for the simple language that we work with, and has a PSPACE upper bound. The high complexity is mainly due to the presence of disjunctions, but we rarely need an unbounded number of disjunctions. When we bound the number of disjunctions, the term derivability problem is in polynomial time. We also explore the complexity of security verification for the active intruder case, and provide a PSPACE upper bound for protocols with boundedly many sessions, with an NP upper bound when the number of disjunctions is bounded as well.

2 Model

2.1 The term model

Fix countable sets Ag, $\mathscr{N}$ and $\mathscr{K}$, denoting the set of *agents*, *nonces* and *keys*, respectively. The set of *basic terms* is $\mathscr{B} = Ag \cup \mathscr{N} \cup \mathscr{K}$. For each $A, B \in Ag$, assume that $sk(A)$, $pk(A)$ and $k(A, B)$ are keys. Further, each $k \in \mathscr{K}$ 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 $\mathscr{T}$ of Dolev-Yao terms is given by the following syntax (where $m \in \mathscr{B}$ and $k \in \mathscr{K}$):

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

For $X \subseteq_{\text{fin}} \mathscr{T}$, we define $\overline{X}$, the *closure* of X, to be the smallest $Y \subseteq \mathscr{T}$ such that (i) $X \subseteq Y$, (ii) $(t, t') \in Y$ iff $\{t, t'\} \subseteq Y$, (iii) if $\{t, k\} \subseteq Y$ then $\{t\}_k \in Y$, and (iv) if $\{\{t\}_k, inv(k)\} \subseteq Y$ then $t \in Y$. We use the notation $X \vdash_{dy} t$ to denote that $t \in \overline{X}$, and $X \vdash_{dy} T$ to denote that $T \subseteq \overline{X}$, for a set of terms T.

We use $st(t)$ to denote the set of *subterms* of t and $st(X) = \bigcup_{t \in X} st(t)$ in Proposition 4, which is a well-known fact about the basic Dolev-Yao model [18].

Proposition 4. *Given $X \subseteq \mathscr{T}$ and $t \in \mathscr{T}$, it can be decided whether $X \vdash_{dy} t$ in time linear in $|st(X \cup \{t\})|$.*

2.2 The assertion language

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

$$\alpha := m \prec t \mid t = t' \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2$$

where $m \in \mathscr{B}$ and $t, t' \in \mathscr{T}$. The assertion $m \prec t$ is to be read as *m occurs in t*. The set of *subformulas* of a formula α is denoted $sf(\alpha)$.

The proof rules for assertions are presented as **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 two parts. Figure 1 gives the rules

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

Fig. 1: The rules for deriving assertions: propositional fragment

pertaining to propositional reasoning with assertions. The rules capture basic reasoning with conjunction and disjunction, and $\perp$ is a restricted contradiction rule.

We next present the rules for atomic assertions of the form $m \prec t$ and $t = t$ in Figure 2. Note that all these rules require X to be nonempty, and some of the rules refer to derivations in the Dolev-Yao theory. For an agent to derive an assertion about a term t, it should know the entire structure of t, which is modelled by saying that from X one can learn (in the Dolev-Yao theory) all basic terms 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. The agent can now derive $m \prec t_{1-i}$ from $m \prec (t_0, t_1)$.

$\dfrac{X \vdash_{dy} m}{X, \Phi \vdash m \prec m}\ ax_2$	$\dfrac{X \vdash_{dy} st(t) \cap \mathscr{B}}{X, \Phi \vdash t = t}\ eq$
$\dfrac{X \vdash_{dy} \{t\}_k \quad X \vdash_{dy} k \quad X, \Phi \vdash m \prec t}{X, \Phi \vdash m \prec \{t\}_k}\ enc$	$\dfrac{X \vdash_{dy} inv(k) \quad X, \Phi \vdash m \prec \{t\}_k}{X, \Phi \vdash m \prec t}\ dec$
$\dfrac{X \vdash_{dy} (t_0, t_1) \quad X, \Phi \vdash m \prec t_i \quad X \vdash_{dy} st(t_{1-i}) \cap \mathscr{B}}{X, \Phi \vdash m \prec (t_0, t_1)}\ pair$	
$\dfrac{X, \Phi \vdash m \prec (t_0, t_1) \quad X \vdash_{dy} st(t_i) \cap \mathscr{B} \quad m \notin st(t_i)}{X, \Phi \vdash m \prec t_{1-i}}\ split$	

Fig. 2: The rules for atomic assertions

We denote by $X, \Phi \vdash_{alp} \alpha$ (resp. $X, \Phi \vdash_{alat} \alpha$; $X, \Phi \vdash_{al} \alpha$) the fact that there is a derivation of $X, \Phi \vdash \alpha$ using the rules in Figure 1 (resp. ax_1 and the rules in Figure 2; the rules in Figures 1 and 2).

2.3 The protocol model

Protocols are typically specified as sequences of communications, but in formal analysis, it is convenient to consider a protocol Pr as a pair $(const, R)$ where $const \subseteq \mathscr{B}$ is a set of constants of Pr and R is a finite set of *roles*. For an agent A, an A-role is a sequence of A-actions. A-actions include send actions of the form $A!B: [(M)t, \{\alpha\}_{sd(A)}]$, and receive actions of the form $A?B: [t, \{\alpha\}_{sd(B)}]$. Here $B \in Ag$, $t \in \mathscr{T}$, $M \subseteq \mathscr{N}$, $\alpha \in \mathscr{A}$, and $\{\alpha\}_{sd(A)}$ denotes the assertion α signed by A. In the send action above, B is merely the intended recipient, and in the receive action, B is merely the purported sender, since we assume the presence of an intruder who can block or forge messages, and can see every communication on the network. For simplicity, we assume that all send and receive actions deal with one term and one assertion. $(M)t$ denotes that the set M contains the nonces used in t, in the context of a protocol run, which are *fresh*, i.e., not used till that point in the run. An additional detail is that *assertions* in sends are always signed by the actual sender, and assertions in receives are signed by the purported sender. Thus, when the intruder I sends an assertion $\{\alpha\}_{sd(A)}$ to someone, it is either replaying an earlier communication from A, or $A = I$ and it can construct α.

We admit two other types of actions in our model, *confirm* and *deny*, to capture conditional branching in protocol specifications. An agent A might, at some stage in a protocol, perform action a_1 if a condition is verified to be true, and a_2 otherwise. For simplicity, we let any assertion be a condition. The behaviour of A in a protocol, in the presence of a branch on condition α, is represented by two sequences of actions, one in which A confirms α and one in which it denies α. These actions are denoted by $A : \mathit{confirm}\ \alpha$ and $A : \mathit{deny}\ \alpha$.

A *knowledge state ks* is of the form $((X_A)_{A \in Ag}, \Phi, SD)$. Here, $X_A \subseteq \mathscr{T}$ is the set of terms accumulated by A in the course of a protocol run till the point under consideration. $\Phi \subseteq \mathscr{A}$ is the set of assertions that have been communicated by any agent (and stored by the intruder). Similarly, $SD \subseteq \{\{\alpha\}_{sd(B)} \mid B \in Ag, \alpha \in \mathscr{A}\}$ is the set of signed assertions communicated by any agent and stored by the intruder.

The *initial knowledge state* of a protocol Pr is $((X_A)_{A \in Ag}, \emptyset, \emptyset)$ where, for each A, $X_A = const(Pr) \cup Ag \cup \{sk(A)\} \cup \{pk(B), k(A, B) \mid B \in Ag\}$.

Let $ks = ((X_A)_{A \in Ag}, \Phi, SD)$ and $ks' = ((X'_A)_{A \in Ag}, \Phi', SD')$ be two knowledge states and a be a send or a receive action. We now describe the conditions under which the execution of a changes ks to ks', denoted $ks \xrightarrow{a} ks'$.

- $a = A!B: [(M)t, \{\alpha\}_{sd(A)}]$
  - a is enabled at ks iff
    - $M \cap X_C = \emptyset$ for all C,
    - $X_A \cup M \vdash_{dy} t$, and
    - $X_A \cup M, \emptyset \vdash_{al} \alpha$.
  - $ks \xrightarrow{a} ks'$ iff
    - $X'_A = X_A \cup M$, $X'_I = X_I \cup \{t\}$,
    - $\Phi' = \Phi \cup \{\alpha\}$, and
    - $SD' = SD \cup \{\{\alpha\}_{sd(A)}\}$.
- $a = A?B: [t, \{\alpha\}_{sd(B)}]$
  - a is enabled at ks iff

    * $\{\alpha\}_{sd(B)}$ is verified as signed by B,
    * $X_I \vdash_{dy} t$, and
    * either $B = I$ and $X_I, \Phi \vdash_{al} \alpha$, or $\{\alpha\}_{sd(B)} \in SD$.
  - $ks \xrightarrow{a} ks'$ iff
    * $X'_A = X_A \cup \{t\}$.
- $a = A : \mathit{confirm}\ \alpha$ is enabled at ks iff $X_A, \emptyset \vdash_{al} \alpha$, and $ks \xrightarrow{a} ks'$ iff $ks = ks'$.
- $a = A : \mathit{deny}\ \alpha$ is enabled at ks iff $X_A, \emptyset \nvdash_{al} \alpha$, and $ks \xrightarrow{a} ks'$ iff $ks = ks'$.

We see that when an agent A sends an assertion α, the intruder stores α in its set of assertions, as well as storing $\{\alpha\}_{sd(A)}$ for possible replay, but honest agents only get to send assertions that they themselves generate from scratch.

A substitution σ is a homomorphism on $\mathscr{T}$ such that $\sigma(Ag) \subseteq Ag$, $\sigma(\mathscr{N}) \subseteq \mathscr{N}$ and $\sigma(\mathscr{K}) \subseteq \mathscr{K}$. σ is said to be suitable for a protocol $Pr = (const, R)$ if $\sigma(m) = m$ for all $m \in const$.

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 $0 \leq lp \leq |\eta|$. $ri = (\eta, \sigma, 0)$ 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 $ri = (\eta, \sigma, lp)$, $ri + 1 = (\eta, \sigma, lp + 1)$. If $ri = (\eta, \sigma, lp)$, $lp \geq 1$ 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 $S' = (S \setminus \{ri\}) \cup \{ri + 1\}$.

A **protocol state** of Pr is a pair $s = (ks, S)$ where ks is a knowledge state of Pr and $S \subseteq RI(Pr)$. $s = (ks, S)$ is an initial protocol state if ks is initial and $S \subseteq IRI(Pr)$. 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 $act(ri + 1) = a$ and $S \xrightarrow{ri} S'$, 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 $TS(Pr)$.

2.4 Comments on the transition rules

The rules for transitioning from a knowledge state, ks, to another, ks', on an action a, deserve some explanation. The change pertaining to the X_A is easily justified by an operational model in which the intruder can snoop on the entire network, but agents are allowed to send only messages they generate. We have extended the same logic to assertions as well, but there is the extra complication of signing the assertions. The intruder typically has access only to its own signature. Thus we posit that the intruder can *replay* assertions signed by another agent A (in case it is passing something in A's name), or that it can generate assertions and sign them in its own name.

As an operational justification for why the honest agents cannot use assertions sent by other agents, 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 can thus produce it whenever demanded. Thus the intruder gets to store α in its local database.

2.5 Example protocol

Recall Example 2 and the attack on it from Section 1.2, reproduced below. We formalize the attack using the transition rules given in Section 2.3.

– $A \rightarrow B : \{m\}_{pk(B)}$
– $B \rightarrow C : \{m\}_{pk(B)}, \{(a \prec \{m\}_{pk(B)}) \vee (b \prec \{m\}_{pk(B)})\}_{sd(B)}$

The attack is informally presented below and is formalized in Figure 3.

1. $A \rightarrow B : \{m\}_{pk(B)}$
2. $B \rightarrow C : \{m\}_{pk(B)},\ \ \{(a \prec \{m\}_{pk(B)}) \vee (r \prec \{m\}_{pk(B)})\}_{sd(B)}$
3. $I \rightarrow B : \{m\}_{pk(B)}$
4. $B \rightarrow C : \{m\}_{pk(B)},\ \ \{(a \prec \{m\}_{pk(B)}) \vee (r' \prec \{m\}_{pk(B)})\}_{sd(B)}$

η_1 : A-role, $\sigma_1(m) = 1$
a_{11} : $A!B$: $\{m\}_{pk(B)}$
η_2 : B-role, $\sigma_2(m) = 1$, $\sigma_2(a) = 1$, $\sigma_2(r) = 3$
a_{21} : $B?A$: $\{m\}_{pk(B)}$
a_{22} : $B!C$: $\{m\}_{pk(B)}$, $\{(a \prec \{m\}_{pk(B)}) \vee (r \prec \{m\}_{pk(B)})\}_{sd(B)}$
η_3 : C-role, $\sigma_3(m) = 1$, $\sigma_3(a) = 1$, $\sigma_3(r) = 3$
a_{31} : $C?B$: $\{m\}_{pk(B)}$, $\{(a \prec \{m\}_{pk(B)}) \vee (r \prec \{m\}_{pk(B)})\}_{sd(B)}$
η_4 : A-role, $\sigma_4(m) = 1$, $\sigma_4(A) = I$
a_{41} : $A!B$: $\{m\}_{pk(B)}$
η_5 : B-role, $\sigma_5(m) = 1$, $\sigma_5(a) = 1$, $\sigma_5(r) = 2$, $\sigma_5(A) = I$
a_{51} : $B?A$: $\{m\}_{pk(B)}$
a_{52} : $B!C$: $\{m\}_{pk(B)}$, $\{(a \prec \{m\}_{pk(B)}) \vee (r \prec \{m\}_{pk(B)})\}_{sd(B)}$
η_6 : C-role, $\sigma_6(m) = 1$, $\sigma_6(a) = 1$, $\sigma_6(r) = 2$
a_{61} : $C?B$: $\{m\}_{pk(B)}$, $\{(a \prec \{m\}_{pk(B)}) \vee (r \prec \{m\}_{pk(B)})\}_{sd(B)}$

Fig. 3: The attack, formalized

In the formalized attack, note that only the actions a_{22} and a_{52} send assertions. It can be seen by applying the given substitutions to these actions, and using the appropriate update actions from the criteria for the change of the knowledge states (as stated earlier), that at the end of this sequence of actions, $\Phi = \{\alpha \vee \beta, \alpha \vee \gamma\}$, where $\alpha = 1 \prec \{1\}_{pk(B)}$, $\beta = 2 \prec \{1\}_{pk(B)}$ and $\gamma = 3 \prec \{1\}_{pk(B)}$.

The intruder can now use these assertions in Φ to perform disjunction elimination, as illustrated in the following proof, and thereby gain the information that the term sent by A is actually $\{1\}_{pk(B)}$, which was supposed to be 'secret'. Thus we see that this protocol admits a run in which there is a reachable state where the intruder is able to learn a secret. This can be thought of as a violation of safety. We elaborate on this idea of safety checking in Section 4.

$$\dfrac{\dfrac{}{\Phi \vdash \alpha \vee \beta}\,ax_1 \quad \dfrac{}{\Phi, \alpha \vdash \alpha}\,ax_1 \quad \dfrac{\dfrac{}{\Phi, \beta \vdash \alpha \vee \gamma}\,ax_1 \quad \dfrac{}{\Phi, \beta, \alpha \vdash \alpha}\,ax_1 \quad \dfrac{\dfrac{}{\Phi, \beta, , \gamma \vdash \beta}\,ax_1 \quad \dfrac{}{\Phi, \beta, \gamma \vdash \gamma}\,ax_1}{\Phi, \beta, \gamma \vdash \alpha}\,\bot}{\Phi, \beta \vdash \alpha}\,\vee e}{\Phi \vdash \alpha}\,\vee e$$

3 The derivability problem and its complexity

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 $X, \Phi \vdash_{al} \alpha$. In this section, we provide a lower bound and an upper bound for this problem, and also some optimizations to the derivability algorithm.

3.1 Properties of the proof system

The following is a useful property that will be crucially used in the lower bound proof. The proof can be found in [16].

Proposition 5. *$X, \Phi \cup \{\alpha \vee \beta\} \vdash_{al} \delta$ iff $X, \Phi \cup \{\alpha\} \vdash_{al} \delta$ and $X, \Phi \cup \{\beta\} \vdash_{al} \delta$.*

Among the rules, *split*, *dec*, $\wedge e$ and $\vee e$ are the **elimination rules**. The rules ax_1, ax_2, *eq*, *split*, *dec* and $\wedge e$ are the **safe rules**, and the rest are the **unsafe rules**. A **normal derivation** is one where no elimination rule has as its major premise the conclusion of an unsafe rule. The following fundamental theorem is on standard lines, and is provided in [16].

Theorem 6. *If there is a derivation of $X, \Phi \vdash \alpha$ then there is a normal derivation of $X, \Phi \vdash \alpha$.*

The following corollaries easily follow by a simple case analysis on derivations.

Corollary 7. *If π is a normal derivation of $X, \Phi \vdash \alpha$ and if the formula β occurs in π, then $\beta \in sf(\Phi \cup \{\alpha\})$.*

Corollary 8. *If $\varnothing, \Phi \vdash_{al} \alpha$ and Φ consists only of atomic assertions, then there is a derivation of the sequent $\varnothing, \Phi \vdash \alpha$ consisting of only the ax, $\wedge i$, $\vee i$ and $\bot$ rules.*

A set of atomic assertions Φ is said to be *contradictory* if there exist distinct nonces m, n, and a nonce b and key k such that both $m \prec \{b\}_k$ and $n \prec \{b\}_k$ are in Φ. Otherwise Φ is *non-contradictory*.

Corollary 9. *If $\varnothing, \Phi \vdash_{al} \alpha$ and Φ is a non-contradictory set of atomic assertions, then there is a derivation of $\varnothing, \Phi \vdash \alpha$ consisting of only the ax, $\wedge i$ and $\vee i$ rules.*

Definition 10 (Derivability problem). *Given $X \subseteq_{fin} \mathscr{T}$, $\Phi \subseteq_{fin} \mathscr{A}$, $\alpha \in \mathscr{A}$, is it the case that $X, \Phi \vdash_{al} \alpha$?*

We first show that the problem is co-NP-hard, and then go on to provide a PSPACE decision procedure. In fact, the hardness result holds even for the propositional fragment of the proof system (consisting of the rules in Figure 1).

3.2 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 $\emptyset, S_\varphi \vdash \overline{\varphi}$ iff φ is a tautology.

Let $\{p_1, p_2, \dots\}$ be the set of all propositional variables. Fix infinitely many nonces $n_1, n_2, \dots$ 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 $\{p_1, \dots, p_n\}$ is the set of all propositional variables occurring in φ. Then $S_\varphi = \{\overline{p_1 \vee \neg p_1}, \dots, \overline{p_n \vee \neg p_n}\}$.

Lemma 11. *$\emptyset, S_\varphi \vdash_{al} \overline{\varphi}$ iff φ is a tautology.*

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

By repeated appeal to Proposition 5, it is easy to see that $\emptyset, S_\varphi \vdash_{al} \overline{\varphi}$ iff for all valuations v over $\{p_1, \dots, p_n\}$, $\emptyset, S_v \vdash_{al} \overline{\varphi}$. We now show that $\emptyset, S_v \vdash_{al} \overline{\varphi}$ iff $v \models \varphi$. The statement of the lemma follows immediately from this.

- We first show by induction on $\psi \in sf(\varphi)$ that $\emptyset, S_v \vdash_{al} \overline{\psi}$ whenever $v \models \psi$.
  - If $\psi = p_i$ or $\psi = \neg p_i$, then $\emptyset, S_v \vdash_{al} \overline{\psi}$ follows from the ax_1 rule.
  - If $\psi = \psi_1 \wedge \psi_2$, then it is the case that $v \models \psi_1$ and $v \models \psi_2$. But then, by induction hypothesis, $\emptyset, S_v \vdash_{al} \overline{\psi_1}$ and $\emptyset, S_v \vdash_{al} \overline{\psi_2}$. Hence, by using $\wedge i$, it follows that $\emptyset, S_v \vdash_{al} \overline{\psi_1 \wedge \psi_2}$.
  - If $\psi = \psi_1 \vee \psi_2$, then it is the case that either $v \models \psi_1$ or $v \models \psi_2$. But then, by induction hypothesis, $\emptyset, S_v \vdash_{al} \overline{\psi_1}$ or $\emptyset, S_v \vdash_{al} \overline{\psi_2}$. In either case, by using $\vee i$, it follows that $\emptyset, S_v \vdash_{al} \overline{\psi_1 \vee \psi_2}$.
- We now show that if $\emptyset, S_v \vdash_{al} \overline{\varphi}$, then $v \models \varphi$. Suppose $\emptyset, S_v \vdash_{al} \overline{\varphi}$. Since S_v is a non-contradictory set of atomic assertions, by Corollary 8, there is a derivation π of $\emptyset, S_v \vdash \overline{\varphi}$ that consists of only the ax, $\wedge i$ and $\vee i$ rules. We now show by induction that for all subproofs π' of π with conclusion $\emptyset, S_v \vdash \overline{\psi}$ that $v \models \psi$.
  - Suppose the last rule of π' is ax_1. Then $\overline{\psi} \in S_v$, and for some $i \leq n$, $\psi = p_i$ or $\psi = \neg p_i$. It can be easily seen by definition of S_v that $v \models \psi$.
  - Suppose the last rule of π' is $\wedge i$. Then $\overline{\psi} = \overline{\psi_1} \wedge \overline{\psi_2}$, and $\emptyset, S_v \vdash_{al} \overline{\psi_1}$ and $\emptyset, S_v \vdash_{al} \overline{\psi_2}$. Thus, by induction hypothesis, $v \models \psi_1$ and $v \models \psi_2$. Therefore $v \models \psi$.
  - Suppose the last rule of π' is $\vee i$. Then $\overline{\psi} = \overline{\psi_1} \vee \overline{\psi_2}$, and either $\emptyset, S_v \vdash_{al} \overline{\psi_1}$ or $\emptyset, S_v \vdash_{al} \overline{\psi_2}$. Thus, by induction hypothesis, either $v \models \psi_1$ or $v \models \psi_2$. Therefore $v \models \psi$. $\dashv$

Theorem 12. *The derivability problem is co-NP-hard.*

3.3 Upper bound

Fix X_0, Φ_0 and α_0. Let $\mathsf{sf} = sf(\Phi_0 \cup \{\alpha_0\})$, $|\mathsf{sf}| = N$, and st be the set of all terms occurring in all assertions in sf. To check whether $X_0, \Phi_0 \vdash \alpha_0$, we check whether α_0 is in the set $deriv(X_0, \Phi_0) = \{\alpha \in \mathsf{sf} \mid X_0, \Phi_0 \vdash \alpha\}$. Below we describe a general procedure to compute $deriv(X, \Phi)$ for any $X \subseteq \mathsf{st}$ and $\Phi \subseteq \mathsf{sf}$.

For $X \subseteq \mathsf{st}$ and $\Phi \subseteq \mathsf{sf}$, define

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

Lemma 13. *$deriv'(X, \Phi)$ is computable in time polynomial in N.*

Proof Let $Y = \{t \in \mathsf{st} \mid X \vdash_{dy} t\}$. Start with $S = \Phi$ and repeatedly add $\alpha \in \mathsf{sf}$ to S whenever α is the conclusion of a rule *other than* $\vee e$ all of whose premises are in $S \cup Y$. Since there are at most N formulas to add to S, and at each step it takes at most N^2 time to check to add a formula, the procedure runs in time polynomial in N. ⊣

We now present the algorithm to compute $deriv(X, \Phi)$. It is presented as two mutually recursive functions f and g, where $g(X, \Phi)$ captures the effect of one application of $\vee e$ for each formula $\alpha_1 \vee \alpha_2 \in \Phi$, and f iterates g appropriately.

```
function $f(X, \Phi)$
    $S \leftarrow \Phi$
    while $S \neq g(X, S)$ do
        $S \leftarrow g(X, S)$
    end while
    return $S$
end function
```

```
function $g(X, \Phi)$
    $S \leftarrow \Phi$
    for all $\alpha_1 \vee \alpha_2 \in S$ do
        if $\alpha_1 \notin S$ and $\alpha_2 \notin S$ then
            $T \leftarrow \{\beta \in f(X, S \cup \{\alpha_1\})\}$
            $U \leftarrow \{\beta \in f(X, S \cup \{\alpha_2\})\}$
            $S \leftarrow S \cup (T \cap U)$
        end if
    end for
    return $deriv'(X, S)$
end function
```

The following theorem asserts the correctness of the algorithm, and its proof follows from Propositions 20 (Soundness) and 21 (Completeness), presented in Appendix A.

Theorem 14. *For $X \subseteq \mathsf{st}$ and $\Phi \subseteq \mathsf{sf}$, $f(X, \Phi) = deriv(X, \Phi)$.*

3.4 Analysis of the algorithm

The nesting depth of recursion in the function f is at most $2N$. We can therefore show that $f(X, \Phi)$ can be computed in $O(N^2)$ space; the proof idea is presented below.

Modify the algorithm for $deriv(X, \Phi)$ using $3N$ global variables S_i, T_i, U_i (i ranging from 0 to $N - 1$), each a bit vector of length N. The procedures f and g take a third argument i, representing the depth of the call in the call tree of $f(X, \Phi, 0)$. $f(\cdot, \cdot, i)$ and

$g(\cdot, \cdot, i)$ use the variables S_i, T_i, U_i. Further, $f(\cdot, \cdot, i)$ makes calls to $g(\cdot, \cdot, i)$ and $g(\cdot, \cdot, i)$ makes calls to $f(\cdot, \cdot, i + 1)$. Since the nesting depth is at most $2N$, the implicit variables on the call stack for arguments and return values are also $O(N)$ in number, so the overall space used is $O(N^2)$.

Theorem 15. *The derivability problem is in PSPACE.*

3.5 Optimization: bounded number of disjunctions

Since the complexity in the algorithm resides mainly in handling $\vee e$, it is worth considering the problem restricted to p disjunctions (independent of N). In this case, the height of the call tree is bounded by $2p$, and since each $f(\cdot, \cdot, i)$ makes at most N calls to $g(\cdot, \cdot, i)$ and each $g(\cdot, \cdot, i)$ makes at most N calls to $f(\cdot, \cdot, i + 1)$, it follows that the total number of calls to f and g is at most N^{2p}. Since $deriv'$ (used by g) can be computed in polynomial time, we have the following theorem.

Theorem 16. *The derivability problem with bounded number of disjunctions is solvable in PTIME.*

As a finer optimization, $deriv'(X, \Phi)$ can be computed in time $O(N)$ by a graph marking algorithm of the kind presented in [14]. This gives an even better running time for the derivability problem in general.

4 Safety checking

The previous section concentrated on 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 important verification problem to study is to *determine the presence of attacks in a protocol*. An attack is typically a sequence of actions conforming to a protocol, with the intruder actively orchestrating communications of the other principals. Formally, an attack on Pr is a run of $TS(Pr)$ that leads to an undesirable system state. The concept is formalized below.

Definition 17 (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 $s \notin \mathsf{Safe}$ reachable in $TS(Pr)$ from an initial protocol state?*

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

Theorem 18. *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. $\dashv$

5 Extending the assertion language

The assertion language presented in 2.2 used disjunction to achieve transmission of partial knowledge. It should be noted that the assertion language used is not constrained to be the same as that one, and various operators and modalities may be added to achieve other desirable properties. In this section, we demonstrate one such addition, namely the *says* modality.

5.1 Assertion language with *says*

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

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

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 himself has no proof of α, A does. A has, in essence, allowed him to transmit this assertion to other agents.

Figure 4 gives rules for assertions of the form A *says* α. For $\sigma = A_1A_2\cdots A_n$, $\sigma : \alpha$ 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 1, and permit propositional reasoning in a modal context.

Like earlier, we denote by $X, \Phi \vdash_{als} \alpha$ the fact that there is a derivation of $X, \Phi \vdash \alpha$ using ax_1 and the rules in Figure 4. We now amend the notation $X, \Phi \vdash_{al} \alpha$ to denote the fact that there is a derivation of $X, \Phi \vdash \alpha$ using the rules in Figures 1, 2 and 4).

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

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

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

Fig. 4: The rules for *says* assertions

5.2 Protocol model

We now outline the modifications to the protocol model occasioned by the *says* modality. The definitions of actions is extended to accommodate the new assertions. However, only *non-modal* assertions are allowed as testable conditions in the *confirm* and *deny* actions.

As regards the transition rules, the addition of the *says* modality allows us one major departure from the definitions specified earlier in 2.3 – agents can reason nontrivially using received assertions. An agent B, on receiving an assertion α from A, can store $A\ says\ \alpha$ in its state, and use it in further derivations. Thus a knowledge state ks is now a tuple of the form $((X_A, \Phi_A)_{A \in Ag}, SD)$.

Let $ks = ((X_A, \Phi_A)_{A \in Ag}, SD)$ and $ks' = ((X'_A, \Phi'_A)_{A \in Ag}, SD')$ be two knowledge states and a be a send or a receive action. We now describe the conditions under which the execution of a changes ks to ks', denoted $ks \xrightarrow{a} ks'$. (These are minor modifications of the rules presented earlier, but presented in full for the convenience of the reader).

- $a = A!B\colon[(M)t, \{\alpha\}_{sd(A)}]$
  - a is enabled at ks iff
    - $M \cap X_C = \varnothing$ for all C,
    - $X_A \cup M \vdash_{dy} t$, and
    - $X_A \cup M, \Phi_A \vdash_{al} \alpha$.
  - $ks \xrightarrow{a} ks'$ iff
    - $X'_A = X_A \cup M$, $X'_I = X_I \cup \{t\}$,
    - $\Phi'_I = \Phi_I \cup \{\alpha, A\ says\ \alpha\}$, and
    - $SD' = SD \cup \{\{\alpha\}_{sd(A)}\}$.
- $a = A?B\colon[t, \{\alpha\}_{sd(B)}]$
  - a is enabled at ks iff
    - $\{\alpha\}_{sd(B)}$ is verified as signed by B,
    - $X_I \vdash_{dy} t$, and
    - either $B = I$ and $X_I, \Phi_I \vdash_{al} \alpha$, or $\{\alpha\}_{sd(B)} \in SD$.
  - $ks \xrightarrow{a} ks'$ iff
    - $X'_A = X_A \cup \{t\}$, and
    - $\Phi'_A = \Phi_A \cup \{B\ says\ \alpha\}$.
- $a = A : \mathit{confirm}\ \alpha$ is enabled at ks iff $X_A, \varnothing \vdash_{al} \alpha$, and $ks \xrightarrow{a} ks'$ iff $ks = ks'$.

– $a = A : deny\ \alpha$ is enabled at ks iff $X_A, \varnothing \nvdash_{al} \alpha$, and $ks \xrightarrow{a} ks'$ iff $ks = ks'$.

We see that on receipt of an assertion α, honest agents always store $A\ says\ \alpha$ in their state, whereas the intruder is allowed to store α itself (along with $A\ says\ \alpha$).

The rest of the definitions extend without any modification.

5.3 Example protocol

A generates a vote, which it wants principals B and C to agree to, and then send to the trusted third party T. However, A does not want B and C to know exactly what the vote is. If a principal agrees to this vote, it prepends its identifier to the term sent to it, encrypts the whole term with the key it shares with T, and sends it to the next agent. Otherwise it merely sends the original term to the next agent. We show the specification where everyone agrees to the vote. $B : A : \alpha$ denotes $B\ says\ A\ says\ \alpha$, and $t = \{v_T\}_k$.

– $A \rightarrow B : t,\ \ \{(a \prec t) \vee (b \prec t)\}_{sd(A)}$
– $B \rightarrow C : \{(B, t)\}_{k(B,T)},\ \ \{A : \{(a \prec t) \vee (b \prec t)\}\}_{sd(B)}$
– $C \rightarrow T : \{(C, \{(B, t)\}_{k(B,T)}\}_{k(C,T)},\ \ \{B : A : \{(a \prec t) \vee (b \prec t)\}\}_{sd(C)}$
– If the nested term is signed by both B and C, and $v_T = a$ or $v_T = b$, $T \rightarrow A : ack$. Otherwise, $T \rightarrow A : 0$.

We now demonstrate an attack. Suppose there is a session S_1 where a and b take values a_1 and b_1, where B agrees to the vote. Suppose now there is a later session S_2 with a taking value a_1 (or b taking value b_1) again. The intruder can now replay the term from B's message to C in S_2 from S_1, although B might not wish to agree in S_2.

5.4 Derivability problem

The basic properties of derivability, including normalization and subformula property, still holds for the expanded language. The lower bound result also carried over without modification, since the formulas featuring in the proof do not involve the *says* modality at all. As regards the upper bound, the procedure g needs to be modified slightly. The modified version is presented below. The proof of correctness is in the appendix.

```
 1: function $g(X, \Phi)$
 2:     $S \leftarrow \Phi$
 3:     for all $\sigma : (\alpha_1 \vee \alpha_2) \in S$ do
 4:         if $\sigma : \alpha_1 \notin S$ and $\sigma : \alpha_2 \notin S$ then
 5:             $T \leftarrow \{\sigma : \beta \in f(X, S \cup \{\sigma : \alpha_1\})\}$
 6:             $U \leftarrow \{\sigma : \beta \in f(X, S \cup \{\sigma : \alpha_2\})\}$
 7:             $S \leftarrow S \cup (T \cap U)$
 8:         end if
 9:     end for
10:     return $deriv'(X, S)$
11: end function
```

An optimization can also be considered, where the functions f and g are modified to take another argument, σ, which provides the **modal context**. Since an application of $\lor e$ on an assertion $\sigma : (\alpha_1 \lor \alpha_2)$ yields only formulas of the form $\sigma : \beta$ in the conclusion, the function $g(\sigma, \cdot, \cdot, i)$ need only make recursive calls to $f(\sigma, \cdot, \cdot, i + 1)$, concentrating only on assertions with prefix σ. Also $f(\sigma, \cdot, \cdot, i)$ need only make recursive calls to $g(\sigma, \cdot, \cdot, i)$ whenever $\sigma \neq \varepsilon$. This has the advantage that the recursion depth is linearly bounded by the maximum number of disjunctions with the *same prefix*. In summary, it is possible to solve the derivability problem efficiently in practical cases.

6 Conclusions

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.

What we would like to see is to arrive at a 'programming methodology' for the structured use of assertions in protocol specifications. As an instance, consider the fact that in our model terms and assertions are bundled together: we communicate (t, α) where binding them requires the same term t to be used in α. Better structuring would use a quantifier *this* in assertions so that references to terms in assertions are contextually bound to communications. This would ensure that in different instantiations (sessions), the assertion would refer to different concrete terms. A more general approach would involve variables in assertions and scoping rules for their instantiations. This raises interesting technical issues and offers further scope for investigation.

References

1. M. Abadi and R. M. Needham. Prudent engineering practices for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22:6–15, 1996.
2. R. Anderson and R. M. Needham. Robustness principles for public-key protocols. In *Proceedings of CRYPTO '95*, LNCS 963, pages 236–247, 1995.
3. M. Backes, C. Hriţcu and M. Maffei. Type-checking zero-knowledge. In *ACM Conference on Computer and Communications Security*, pages 357–370, 2008.
4. M. Backes, M. Maffei and D. Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *IEEE Symposium on Security and Privacy*, pages 202–215, 2008.
5. A. Baskar, P. Naldurg, K. R. Raghavendra and S. P. Suresh. Primal Infon Logic: Derivability in Polynomial Time. In *Proceedings of FSTTCS 2013*, LIPIcs 24, pages 163–174, 2013.

6. A. Baskar, R. Ramanujam and S.P. Suresh. A DEXPTIME-complete Dolev-Yao theory with distributive encryption. In *Proceedings of MFCS 2010*, LNCS 6281, pages 102–113, 2010.
7. M. Burrows, M. Abadi and R. M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, Feb 1990.
8. J. Benaloh. Cryptographic capsules: A disjunctive primitive for interactive protocols. In *Proceedings of CRYPTO '86*, LNCS 263, pages 213–222, 1987.
9. H. Comon and V. Shmatikov. Intruder Deductions, Constraint Solving and Insecurity Decisions in Presence of Exclusive or. In *Proceedings of LICS 2003*, pages 271–280, June 2003.
10. V. Cortier, S. Delaune and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.
11. S. Delaune, S. Kremer and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.
12. D. Dolev and A. Yao. On the Security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.
13. G. Fuchsbauer and D. Pointcheval. Anonymous consecutive delegation of signing rights: Unifying group and proxy signatures. In *Formal to Practical Security*, pages 95–115, 2009.
14. Y. Gurevich and I. Neeman. Infon logic: the propositional case. *ACM Transactions on Computational Logic*, 12(2):9:1–9:28, 2011.
15. P. Lafourcade, D. Lugiez and R. Treinen. Intruder deduction for the equational theory of abelian groups with distributive encryption. *Information and Computation*, 205(4):581–623, April 2007.
16. R. Ramanujam, V. Sundararajan and S.P. Suresh. Extending Dolev-Yao with assertions. Technical Report. 2014. URL: `http://www.cmi.ac.in/~spsuresh/dyassert.pdf`.
17. Zuzana Rjaskova. Electronic voting schemes. Master's Thesis, Comenius University. 2002.
18. M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.

Appendix

A Algorithm for derivability: correctness proof

For a fixed X, define $f_X : \wp(\mathsf{sf}) \to \wp(\mathsf{sf})$ to be the function that maps Φ to $f(X, \Phi)$. Similarly, $g_X(\Phi)$ is defined to be $g(X, \Phi)$.

Lemma 19. *1.* $\Phi \subseteq deriv'(X, \Phi) \subseteq deriv(X, \Phi)$.
2. $deriv'(X, deriv(X, \Phi)) = deriv(X, deriv(X, \Phi)) = deriv(X, \Phi)$.
3. *If* $\Phi \subseteq \Psi$ *then* $g_X(\Phi) \subseteq g_X(\Psi)$ *and* $f_X(\Phi) \subseteq f_X(\Psi)$.
4. $\Phi \subseteq g_X(\Phi) \subseteq g_X^2(\Phi) \subseteq \cdots \subseteq \mathsf{sf}$.
5. $f_X(\Phi) = g_X^m(\Phi)$ *for some* $m \leq N$.

The last fact is true because $|\mathsf{sf}| = N$ and the $g_X^i(\Phi)$s form a non-decreasing sequence.

Proposition 20 (Soundness). *For* $X \subseteq \mathsf{st}$, $\Phi \subseteq \mathsf{sf}$ *and* $m \geq 0$, $g_X^m(\Phi) \subseteq deriv(X, \Phi)$.

Proof We shall assume that

$$g_X^n(\Psi) \subseteq deriv(X, \Psi) \text{ for all } \Psi \subseteq \mathsf{sf}, n \geq 0 \text{ s.t. } (N - |\Psi|, n) <_{\text{lex}} (N - |\Phi|, m)$$

and prove that

$$g_X^m(\Phi) \subseteq deriv(X, \Phi).$$

Now if $m = 0$, then $g_X^m(\Phi) = \Phi \subseteq deriv(X, \Phi)$. Suppose $m > 0$, Let $Z = g_X^{m-1}(\Phi)$ and let $S \subseteq \mathsf{sf}$ be such that $\alpha \in S$ iff one of the following conditions hold:

- $\alpha \in Z$
- α is of the form $\sigma : \beta$ and there is some $\sigma : (\alpha_1 \vee \alpha_2) \in Z$ such that $\sigma : \alpha_i \notin Z$ and $\alpha \in f_X(Z \cup \{\sigma : \alpha_1\}) \cap f_X(Z \cup \{\sigma : \alpha_2\})$.

Observe that since $(N - |\Phi|, m - 1) <_{\text{lex}} (N - |\Phi|, m)$, by induction hypothesis, $Z = g_X^{m-1}(\Phi) \subseteq deriv(X, \Phi)$. To conclude that $g_X^m(\Phi) \subseteq deriv(X, \Phi)$, it suffices to prove that $S \subseteq deriv(X, \Phi)$, since then we have

$$g_X^m(\Phi) = deriv'(X, S) \subseteq deriv'(X, deriv(X, \Phi)) = deriv(X, \Phi).$$

Now if $\alpha \in S$, then there are two cases:

- $\alpha \in Z$. But $Z \subseteq deriv(X, \Phi)$, and so $\alpha \in deriv(X, \Phi)$.
- α is of the form $\sigma : \beta$ and $\alpha \in f_X(Z \cup \{\sigma : \alpha_1\}) \cap f_X(Z \cup \{\sigma : \alpha_2\})$ for some $\sigma : (\alpha_1 \vee \alpha_2) \in Z$. For any Ψ, $f_X(\Psi) = g_X^n(\Psi)$ for some $n \leq N$, and for any n, $(N - |Z \cup \{\sigma : \alpha_i\}|, n) <_{\text{lex}} (N - |\Phi|, m)$. Thus, by induction hypothesis, $f_X(Z \cup \{\sigma : \alpha_i\}) \subseteq deriv(X, Z \cup \{\sigma : \alpha_i\})$. In other words, $X, Z \cup \{\sigma : \alpha_1\} \vdash_{al} \sigma : \beta$ and $X, Z \cup \{\sigma : \alpha_2\} \vdash_{al} \sigma : \beta$ and $X, Z \vdash_{al} \sigma : (\alpha_1 \vee \alpha_2)$. By an application of the $\vee e$ rule, we conclude that $X, Z \vdash_{al} \sigma : \beta$. Thus

$$\alpha \in deriv(X, Z) \subseteq deriv(X, deriv(X, \Phi)) = deriv(X, \Phi).$$

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

Proposition 21 (Completeness). *For $X \subseteq \mathsf{st}$, $\Phi \subseteq \mathsf{sf}$ and $\alpha \in deriv(X, \Phi)$, there is $m \geq 0$ such that $\alpha \in g_X^m(\Phi)$.*

Proof Suppose $\alpha \in deriv(X, \Phi)$. Then there is a normal derivation π of $X, \Phi \vdash \alpha$. We now prove the desired claim by induction on the structure of π.

- Suppose the last rule r of π is not $\vee e$. If r is ax_1, $\alpha \in \Phi = g_X^0(\Phi)$. If not, let $S = \{\beta \mid X, \Phi \vdash \beta \text{ is a premise of } \mathsf{r}\}$. Since each $\beta \in S$ is the conclusion of a subproof of π, by induction hypothesis, there is an m such that $\beta \in g_X^m(\Phi)$. It follows that there is n such that $S \subseteq g_X^n(\Phi)$. Since for any Ψ, $deriv'(X, \Psi) \subseteq g_X(\Psi)$, it follows that $\alpha \in deriv'(X, S) \subseteq deriv'(X, g_X^n(\Phi)) \subseteq g_X^{n+1}(\Phi)$.
- 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 subproofs of π with conclusions $X, \Phi \vdash \sigma : (\alpha_1 \vee \alpha_2)$, $X, \Phi \cup \{\sigma : \alpha_1\} \vdash \sigma : \beta$ and $X, \Phi \cup \{\sigma : \alpha_2\} \vdash \sigma : \beta$. By induction hypothesis, there are m, n, p such that $\sigma : (\alpha_1 \vee \alpha_2) \in g_X^m(\Phi)$, $\sigma : \beta \in g_X^n(\Phi \cup \{\alpha_1\})$ and $\sigma : \beta \in g_X^p(\Phi \cup \{\alpha_2\})$. Since $g_X^q(\Psi) \subseteq f_X(\Psi)$ for any Ψ and $q \geq 0$, it follows that $\sigma : \beta \in f_X(\Phi \cup \alpha_1\}) \cap f_X(\Phi \cup \alpha_2\})$. Thus $\sigma : \beta \in g_X^{m+1}(\Phi)$. $\dashv$