Extending Dolev-Yao with assertions[*]

R. Ramanujam
IMSc, Chennai
jam@imsc.res.in

Vaishnavi Sundararajan
CMI, Chennai
vaishnavi@cmi.ac.in

S.P. Suresh
CMI, Chennai
spsuresh@cmi.ac.in

April 22, 2014

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 [13], 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

Formal verification of security properties of cryptographic protocols requires an abstract model of agents' capabilities and communications, and the **Dolev-Yao model** [13] 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. Typically extensions cater to more complex cryptographic operations like homomorphic encryption, blind signatures etc, that are useful in applications like e-voting and contract signing [17, 6, 12]. The interaction between the operators can have significant impact on term derivability, and forms an important line of theoretical research [10, 17, 11].

[*]We thank A. Baskar for discussions and comments on many ideas in the paper.

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. On the other hand, cryptographic protocols often use *certificates* that can be verified but cannot be "owned". For instance, an agent A may wish to certify that an encrypted term contains a 0 or a 1 to another agent who cannot decrypt it. The recipient cannot generate such a certificate but can only further certify it as having been received from A and pass it along.

In this paper, we suggest that such certification can be seen as an extension of the Dolev-Yao model in which agents have the ability to communicate *assertions*. This ability is a common feature in distributed protocols in general: agents can communicate not just data, but also assertions about their local state or about previously communicated data. Communicating both data and assertions gives significant flexibility for formal specification and reasoning about such protocols, and we suggest that it is worthwhile in the context of security protocols as well. Note that the idea of communicating assertions is already found in BAN logic [8] and related papers. But they translate all communications to assertions (the 'idealisation' step) and use a rich language with belief modalities.

It should be clear that there is no real *need* for such an extension. After all, assertions are merely a kind of data, and a type discipline on communicated terms can disambiguate data and assertions. Indeed, in the cryptographic setting, assertions are often encoded as terms during the protocol design (often by adding new constructors to the term algebra), and are reasoned about via the encoding terms [3]. However, 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 assertions, where the sender constructs a proof and sends it, which the receiver then verifies [4]. While such an approach is natural in models that view assertions as data, treating them as distinct offers a crucial point of departure via an abstraction similar to the perfect encryption assumption in the Dolev-Yao model. We could 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. This can be done without restricting intruder capabilities greatly.

What constitutes an appropriate logical language of assertions is an interesting body of research in itself. Many logics have been based on the seminal BAN logic [8], which express security properties that can be formally checked in protocols or derived as theorems in appropriate systems. In principle, any such logical language can be used as an assertion language for the extension we propose to the Dolev-Yao model as well. However, this would go against the spirit of the proposal here. Much as the Dolev-Yao model is principally a minimal term algebra with its derivation rules, we consider the extension with assertions as again a minimal logic with its associated derivation rules. What we propose here is meant not as a specific extension with a particular assertion language that we advocate, but as studying the implications of such extension (for the passive and active intruder cases), illustrated by a specific assertion language.

We suggest that a propositional modal language, with highly restricted use of negation and modalities, is appropriate. In this sense, our inspiration is *infon logic* [16, 5] for reasoning about access control. A priori, certification in cryptographic protocols reveals partial knowledge 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, $(0 \textit{ occurs in } t) \lor (1 \textit{ occurs in } t)$ can be seen as a *partial secrecy assertion*. Note that background knowledge of the Dolev-Yao model offers implicit atomic negation: $0 \textit{ occurs in } \{m\}_k$ where m is atomic may exclude the assertion $1 \textit{ occurs in } \{m\}_k$. With conjunctions to bundle assertions together, we have a restricted propositional language.

The modality we study is one that refers to agents passing certificates along, and has the flavour of *delegation*: A sending α to B allows B to send $(A \textit{ says } \alpha)$ 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, [9] presents a new cryptographic primitive for partial information transmission, while [15] deals with delegation and signatures, although there the focus is more on anonymity and group signatures.

We show that even such minimal propositional assertion languages (with no negation, no implication, and seemingly positive, but for the implicit negation forced by the term model) pack a punch: the term derivability problem is co-NP-hard. Note that such a result critically pertains to the underlying Dolev-Yao model since positive logics are not, in general, computationally hard to reason about. A theoretical observation of this kind can be mitigated by practical considerations: in protocols, we rarely need an unbounded number of disjunctions (or partial secrecy assertions), and in such cases, the term derivability problem is in polynomial time.

The proof system that we employ is an extension of the positive fragment of propositional intuitionistic logic, with some rules for atomic assertions, and a restricted contradiction rule. For the modalities, we permit propositional reasoning inside contexts, but do not allow distribution of the modalities over other operators. For example, one can derive $A \textit{ says } (\alpha \lor \beta)$ from $A \textit{ says } \alpha$, but one may not derive $(A \textit{ says } \alpha) \lor (A \textit{ says } \beta)$ from $A \textit{ says } (\alpha \lor \beta)$.

Our work is principally a proof theoretic investigation of passive intruder capabilities in an extension of the Dolev-Yao model with assertions, but we also illustrate the use of these assertions for exploring active intruder attacks. We explore the complexity of security verification for the active intruder case, and it is on expected lines. We 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 Assertions and derivations

2.1 Dolev-Yao 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 $k \in \mathscr{K}$):

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

Definition 1 *For $X \subseteq_{fin} \mathscr{T}$, we define $\overline{X}$, the* closure *of X, to be the smallest $Y \subseteq \mathscr{T}$ such that:*

- $X \subseteq Y$,
- $(t, t') \in Y$ *iff* $\{t, t'\} \subseteq Y$,
- *if* $\{t, k\} \subseteq Y$ *then* $\{t\}_k \in Y$, *and*
- *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.

The following fact about the basic Dolev-Yao model is well known [19]. (We use $st(t)$ to denote the set of *subterms* of t and $st(X) = \bigcup_{t \in X} st(t)$, in Proposition 2.)

Proposition 2 *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 Assertion language

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

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

where $m \in \mathscr{B}$ and $t, t' \in \mathscr{T}$. The assertion $m \prec t$ is 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 in terms of **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. Figure 1 gives the rules 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 $m \prec (t_0, t_1)$.

Figure 3 gives rules for assertions of the form A *says* α. For $\sigma = A_1 A_2 \cdots A_n$, we let $\sigma : \alpha$ denote A_1 *says* $(A_2$ *says* $\cdots (A_n$ *says* $\alpha) \cdots)$, and define $\sigma : \Phi$ to be $\{\sigma : \alpha \mid \alpha \in \Phi\}$. These rules are direct generalizations of the propositional rules in Figure 1, and permit propositional reasoning in a modal context.

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 1 (resp. ax_1 and the rules in Figure 2; ax_1 and the rules in Figure 3; the rules in Figures 1, 2 and 3).

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

Figure 1: The rules for deriving assertions: propositional fragment

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

Figure 2: The rules for atomic assertions

2.3 Properties of the proof system

Proposition 3 1. *If $X, \Phi \vdash_{alat} \sigma : \alpha$ then $\sigma = \varepsilon$.*

2. *If $X, \Phi \vdash_{als} \sigma : \alpha$ and $\sigma \neq \varepsilon$, then $\emptyset, \sigma : \Phi' \vdash_{als} \sigma : \alpha$, where $\sigma : \Phi' \subseteq \Phi$.*

3. ***Monotonicity*** *If $X, \Phi \vdash_{al} \alpha$, $X \subseteq X'$ and $\Phi \subseteq \Phi'$, then $X', \Phi' \vdash_{al} \alpha$.*

4. ***Cut*** *If $X, \Phi \vdash_{al} \alpha$ and $X, \Phi \cup \{\alpha\} \vdash_{al} \beta$, then $X, \Phi \vdash_{al} \beta$.*

Corollary 4 *$X, \Phi \cup \{\alpha \vee \beta\} \vdash_{al} \delta$ iff $X, \Phi \cup \{\alpha\} \vdash_{al} \delta$ and $X, \Phi \cup \{\beta\} \vdash_{al} \delta$.*

Proof

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

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

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

Figure 3: The rules for *says* assertions

($\Leftarrow$) Suppose π is a derivation of $X, \Phi \cup \{\alpha \vee \beta, \alpha\} \vdash \delta$ and π' is a derivation of $X, \Phi \cup \{\alpha \vee \beta, \beta\} \vdash \delta$. Then the following is a derivation of $X, \Phi \cup \{\alpha \vee \beta\} \vdash \delta$.

$$\dfrac{\dfrac{}{X,\Phi \cup \{\alpha \vee \beta\} \vdash \alpha \vee \beta}\ ax \qquad \begin{matrix}\pi \\ \vdots \\ X,\Phi \cup \{\alpha \vee \beta, \alpha\} \vdash \delta\end{matrix} \quad \begin{matrix}\pi' \\ \vdots \\ X,\Phi \cup \{\alpha \vee \beta, \beta\} \vdash \delta\end{matrix}}{X,\Phi \cup \{\alpha \vee \beta\} \vdash \delta}\ \vee e$$

($\Rightarrow$) There are simple derivations of $X, \Phi \cup \{\alpha\} \vdash \alpha \vee \beta$ and $X, \Phi \cup \{\beta\} \vdash \alpha \vee \beta$. Therefore if $X, \Phi \cup \{\alpha \vee \beta\} \vdash_{al} \delta$, it follows from the admissibility of Cut that $X, \Phi \cup \{\alpha\} \vdash_{al} \delta$ and $X, \Phi \cup \{\beta\} \vdash_{al} \delta$. $\dashv$

Among the rules, *split*, *dec*, $\wedge e$ and $\vee e$ are the **elimination rules** and the rest are **introduction 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. A derivation is normal iff its **rank** is 0, as given by the following definition.

Definition 5 (Rank of a derivation) *Let π be a derivation with conclusion $X, \Phi \vdash \alpha$ and last rule r. 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 an elimination rule and r_1 is an unsafe rule, then*

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

- *Otherwise*

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

Proposition 6 *If π is a derivation of $X, \Phi \vdash \alpha$ and π' a derivation of $X, \Phi \cup \{\alpha\} \vdash \beta$, then there is a derivation ϖ of $X, \Phi \vdash \beta$ such that*

$$rank(\varpi) \leq \max(rank(\pi), rank(\pi'), |\alpha|).$$

Proof Replace all sequents of the form $X, \Phi \cup \Psi \cup \{\alpha\} \vdash \gamma$ in π' by $X, \Phi \cup \Psi \vdash \gamma$. To make this a valid derivation, replace all occurrences of $X, \Phi \cup \Psi \vdash \alpha$ at the leaf level by the derivation π. This is the derivation ϖ. Suppose π ends in an unsafe rule, and the axiom $X, \Phi \cup \Psi \cup \{\alpha\} \vdash \alpha$ in π' is a major premise of an elimination rule. The contribution of the axiom to the rank of this subproof of ϖ is $\max(rank(\pi), |\alpha|)$. However, the minor premise(s) of the rule could have rank(s) as high as $rank(\pi')$, and therefore, the rank of this particular subproof of ϖ is at most $\max(rank(\pi), rank(\pi'), |\alpha|)$. Since the structure of π' is otherwise preserved in ϖ, it follows that $rank(\varpi) \leq \max(rank(\pi), rank(\pi'), |\alpha|)$. $\dashv$

Lemma 7 *Suppose π is a derivation with conclusion $X, \Phi \vdash \alpha$ and last rule* r *such that $rank(\pi) = d > 0$, and all proper subderivations of π are of rank $< d$. Then the following hold.*

1. *If* r *is not $\vee e$, then $|\alpha| < d$.*
2. *There is a derivation π' of $X, \Phi \vdash \alpha$ such that $rank(\pi') < d$.*

Proof Suppose the major premise of r is $X, \Phi \vdash \beta$, the conclusion of π_1 ending with rule r_1. Given the conditions of the lemma, it is clear that $rank(\pi) = |\beta|$, r_1 is unsafe, and r is an elimination rule.

1. If r is not $\vee e$, then we have the following cases:
   - $\beta = m \prec (t, t')$ and $\alpha = m \prec t$.
   - $\beta = m \prec \{t\}_k$ and $\alpha = m \prec t$.
   - $\beta = \sigma : (\gamma \wedge \gamma')$ and $\alpha = \sigma : \gamma$.

   In all these cases, it is clear that $|\alpha| < |\beta| = d$.

2. To show the existence of π', we perform a case analysis on r_1 and r, and prove the result by induction on $||\pi_1|| + ||\pi_2||$, where $||\pi||$ is the size of the proof π.
   - Suppose r is not $\vee e$ and r_1 is the introduction rule corresponding to r. Then π' is one of the immediate subproofs of π_1, and clearly $rank(\pi') < d$.
   - Suppose r is $\vee e$ and r_1 is $\vee i$. Say $\beta = \beta_1 \vee \beta_2$. Then there is an immediate subproof (say π_2) of π with conclusion $X, \Phi \cup \{\beta_1\} \vdash \alpha$. There is also an immediate subproof π_{11} of π_1 with conclusion $X, \Phi \vdash \beta_1$. Thus we can apply cut on π_{11} and π_2 to get a derivation of $X, \Phi \vdash \alpha$ whose rank is $\max(|\beta_1|, rank(\pi_{11}), rank(\pi_2)) < d$.
   - Suppose r_1 is $\perp$. Then we can replace the conclusion of π_1 directly by $X, \Phi \vdash \alpha$.
   - Suppose r_1 is $\vee e$. Suppose, for example, r is $\wedge e$. Then π has the following form:

$$\dfrac{\dfrac{\begin{matrix}\pi_{11}\\ \vdots \\ X, \Phi \vdash \gamma_1 \vee \gamma_2\end{matrix} \quad \begin{matrix}\pi_{12}\\ \vdots \\ X, \Phi \cup \{\gamma_1\} \vdash \beta\end{matrix} \quad \begin{matrix}\pi_{13}\\ \vdots \\ X, \Phi \cup \{\gamma_2\} \vdash \beta\end{matrix}}{X, \Phi \vdash \beta}\vee e}{X, \Phi \vdash \alpha}\wedge e$$

This can be transformed to the following proof π':

$$\dfrac{\begin{array}{c}\pi_{11}\\ \vdots\\ X, \Phi \vdash \gamma_1 \vee \gamma_2\end{array} \quad \dfrac{\begin{array}{c}\pi_{12}\\ \vdots\\ X, \Phi \cup \{\gamma_1\} \vdash \beta\end{array}}{X, \Phi \cup \{\gamma_1\} \vdash \alpha}\wedge e \quad \dfrac{\begin{array}{c}\pi_{13}\\ \vdots\\ X, \Phi \cup \{\gamma_2\} \vdash \beta\end{array}}{X, \Phi \cup \{\gamma_2\} \vdash \alpha}\wedge e}{X, \Phi \vdash \alpha}\vee e$$

Since $\|\pi_{12}\| < \|\pi_1\|$ and $\|\pi_{13}\| < \|\pi_1\|$, we can appeal to induction hypothesis to get proofs π_2' and π_3' of rank $< d$, which when composed with π_{11} in the above manner, yields a proof π' of rank $< d$. $\dashv$

Theorem 8 (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 $d = rank(\pi)$, and n is the number of subderivations of π of rank d. If $rank(\pi) = 0$, π is already normal. If not, let $rank(\pi) = d > 0$ and let ϖ be a subderivation of π with conclusion $X, \Psi \vdash \beta$ such that $rank(\varpi) = d$ and no proper subderivation of ϖ is of rank $\geq d$. By Lemma 7, $|\beta| < d$ and there is another derivation ϖ' with $rank(\varpi') < d$ 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 $\geq d$ 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 $\geq d$ 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. $\dashv$

Corollary 9 *If $\emptyset, \Phi \vdash_{al} \alpha$ and Φ consists only of atomic assertions, then there is a derivation of the sequent $\emptyset, \Phi \vdash \alpha$ 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 10 *If $\emptyset, \Phi \vdash_{al} \alpha$ and Φ is a non-contradictory set of atomic assertions, then there is a derivation of $\emptyset, \Phi \vdash \alpha$ consisting of only the ax, $\wedge i$ and $\vee i$ rules.*

Theorem 11 *If π is a normal derivation $X, \Phi \vdash \alpha$ and if a formula β occurs in π, then $\beta \in sf(\Phi \cup \{\alpha\})$. Furthermore, if the last rule of π is an elimination, then $\beta \in sf(\Phi)$.*

Proof Let r be the last rule of π. The claim is obvious when r is an introduction rule. We look at a few cases when it is an elimination rule. Suppose π_1 is the subproof of π whose conclusion, $X, \Phi \vdash \gamma$, is the major premise of r. Since π is normal, the last rule of π_1, say r_1, is not unsafe. So it is either an axiom or an elimination rule other than $\vee e$. Therefore $\alpha \in sf(\gamma)$. Now any β occurring in π either occurs in an immediate subproof or in the sequent $X, \Phi \vdash \alpha$. If it occurs in π_1, then $\beta \in sf(\Phi)$. So, in particular, $\gamma \in sf(\Phi)$, and the same holds for α. If it occurs in some other immediate subproof, then $\beta \in sf(\Phi \cup \{\alpha, \gamma\}) \subseteq sf(\Phi)$ (β could belong to $sf(\gamma)$ if r is $\vee e$). This concludes the proof. $\dashv$

3 Complexity of derivability problem

Definition 12 (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).

3.1 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 $\{p_1, p_2, \ldots\}$ 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 propositional variables occurring in φ. Then $S_\varphi = \{\overline{p_1 \vee \neg p_1}, \ldots, \overline{p_n \vee \neg p_n}\}$.

Lemma 13 $\emptyset, S_\varphi \vdash_{al} \overline{\varphi}$ *iff φ is a tautology.*

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

By repeated appeal to Corollary 4, it is easy to see that $\emptyset, S_\varphi \vdash_{al} \overline{\varphi}$ iff $\emptyset, S_v \vdash_{al} \overline{\varphi}$ for all valuations v over $\{p_1, \ldots, p_n\}$. 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, $\varnothing, S_v \vdash_{al} \overline{\psi_1}$ and $\varnothing, S_v \vdash_{al} \overline{\psi_2}$. Hence, by using $\wedge i$, it follows that $\varnothing, S_v \vdash_{al} \overline{\psi_1 \wedge \psi_2}$.
- 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, $\varnothing, S_v \vdash_{al} \overline{\psi_1}$ or $\varnothing, S_v \vdash_{al} \overline{\psi_2}$. In either case, by using $\vee i$, it follows that $\varnothing, S_v \vdash_{al} \overline{\psi_1 \vee \psi_2}$.

• We now show that if $\varnothing, S_v \vdash_{al} \overline{\varphi}$, then $v \models \varphi$. Suppose $\varnothing, S_v \vdash_{al} \overline{\varphi}$. Since S_v is a non-contradictory set of atomic assertions,by Corollary 10, there is a derivation π of $\varnothing, S_v \vdash \overline{\varphi}$ 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 $\varnothing, S_v \vdash \overline{\psi}$ 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 $\varnothing, S_v \vdash_{al} \overline{\psi_1}$ and $\varnothing, S_v \vdash_{al} \overline{\psi_2}$. 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 $\varnothing, S_v \vdash_{al} \overline{\psi_1}$ or $\varnothing, S_v \vdash_{al} \overline{\psi_2}$. Thus, by induction hypothesis, either $v \models \psi_1$ or $v \models \psi_2$. Therefore $v \models \psi$. $\dashv$

Theorem 14 *The derivability problem is co-NP-hard.*

3.2 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 $deriv'(X, \Phi)$ to be the set of all $\alpha \in \mathsf{sf}$ such that $X, \Phi \vdash \alpha$ is provable by a derivation which does not use the $\vee e$ rule.

Lemma 15 *$deriv'(X, \Phi)$ 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. $\dashv$

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.

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 $g(X, \Phi)$.

Algorithm 1 Algorithm to compute $deriv(X, \Phi)$

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

Lemma 16

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 nondecreasing sequence.

Proposition 17 (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, $Z = g_X^{m-1}(\Phi) \subseteq \mathit{deriv}(X, \Phi)$. To conclude that $g_X^m(\Phi) \subseteq \mathit{deriv}(X, \Phi)$, it suffices to prove that $S \subseteq \mathit{deriv}(X, \Phi)$, since then we have

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

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

- $\alpha \in Z$. But $Z \subseteq \mathit{deriv}(X, \Phi)$, and so $\alpha \in \mathit{deriv}(X, \Phi)$.
- 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, $f_X(Z \cup \{\sigma : \alpha_i\}) \subseteq \mathit{deriv}(X, Z \cup \{\sigma : \alpha_i\})$. 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

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

$\dashv$

Proposition 18 (Completeness) *For $X \subseteq$ **st**, $\Phi \subseteq$ **sf** and $\alpha \in \mathit{deriv}(X, \Phi)$, there is $m \geq 0$ such that $\alpha \in g_X^m(\Phi)$.*

Proof Suppose $\alpha \in \mathit{deriv}(X, \Phi)$. 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 $S = \{\beta \mid X, \Phi \vdash \beta$ 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, $\mathit{deriv}'(X, \Psi) \subseteq g_X(\Psi)$, it follows that $\alpha \in \mathit{deriv}'(X, S) \subseteq \mathit{deriv}'(X, g_X^n(\Phi)) \subseteq g_X^{n+1}(\Phi)$.
- 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 $\sigma : \beta \in g_X^{m+1}(\Phi)$.

$\dashv$

Theorem 19 *For $X \subseteq$ **st** and $\Phi \subseteq$ **sf**, $f_X(\Phi) = deriv(X, \Phi)$.*

3.3 Analysis of the algorithm

Lemma 20 *The nesting depth of recursion in the function f is at most $2N$.*

Proof From the description of the algorithm, it is clear that for all calls $g(X, \Psi)$ from $f(X, \Phi)$ and for all calls $f(X, \Psi)$ from $g(X, \Phi)$, $\Phi \subseteq \Psi$. The statement of the lemma follows immediately. $\dashv$

Lemma 21 *$f(X, \Phi)$ can be computed in $O(N^2)$ space.*

Proof We modify Algorithm 1 using $3N$ global variables $S_i, T_i, U_i (i < N)$, 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, $f(\cdot, \cdot, i)$ makes calls to $g(\cdot, \cdot, i)$, and $g(\cdot, \cdot, i)$ makes calls to $f(\cdot, \cdot, i + 1)$. There are implicit variables on the call stack for arguments and return values but the nesting depth is at most $2N$, so the overall space used is $O(N^2)$. $\dashv$

Theorem 22 *The derivability problem is in* PSPACE.

3.3.1 Bounded number of disjunctions

Since the complexity in the algorithm resides mainly in handling $\vee e$, it is worth considering the problem restricted to K disjunctions (independent of N). In this case, the height of the call tree is bounded by $2K$, 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^{2K}. Since $deriv'$ (used by g) can be computed in polynomial time, we have the following theorem.

Theorem 23 *The derivability problem with bounded number of disjunctions is in* PTIME.

3.3.2 Optimizations

There are several ways in which our algorithm can be improved. As a first observation, $deriv'(X, \Phi)$ can be computed in time $O(N)$ by a graph marking algorithm as presented in 3.3.3 (similar to the algorithm in [16]). Secondly, the functions f and g can be modified to take another argument, σ, which provides the **modal context**. Since an application of $\vee e$ on an assertion $\sigma : (\alpha_1 \vee \alpha_2)$ yields 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.

3.3.3 $O(N)$ algorithm for $\mathit{deriv}'(X, \Phi)$

For a formula $x = \sigma : \alpha$, we define $\mathit{context}(x) = \sigma$. We also define the notions $\mathit{left}(x)$, $\mathit{right}(x)$, and $\mathit{op}(x)$ as follows:

- If $x = \sigma : t = t'$, $\mathit{left}(x) = t$, $\mathit{right}(x) = t'$, and $\mathit{op}(x) = e$.
- If $x = \sigma : m \prec t$, $\mathit{left}(x) = m$, $\mathit{right}(x) = t$, and $\mathit{op}(x) = \prec$.
- If $x = \sigma : y \wedge z$, $\mathit{left}(x) = y$, $\mathit{right}(x) = z$, and $\mathit{op}(x) = \wedge$.
- If $x = \sigma : y \vee z$, $\mathit{left}(x) = y$, $\mathit{right}(x) = z$, and $\mathit{op}(x) = \vee$.

For every $x \in \mathit{sf}$, define the following sets:

- $\mathsf{A}_\ell(x) = \{y \in \mathsf{sf} \mid \mathit{context}(y) = \mathit{context}(x), \mathit{op}(y) = \wedge \text{ and } \mathit{left}(y) = x\}$.
- $\mathsf{A}_r(x) = \{y \in \mathsf{sf} \mid \mathit{context}(y) = \mathit{context}(x), \mathit{op}(y) = \wedge \text{ and } \mathit{right}(y) = x\}$.
- $\mathsf{O}_c(x) = \{y \in \mathsf{sf} \mid \mathit{context}(y) = \mathit{context}(x), \mathit{op}(y) = \vee, \text{ and } \mathit{left}(y) = x \text{ or } \mathit{right}(y) = x\}$.
- $\mathit{child}(x) = \{y \in \mathsf{sf} \mid \mathit{left}(x) = \mathrm{y} \text{ or } \mathit{right}(x) = y\}$.

For every $x \in \mathit{sf}$ of the form $\sigma : a \prec \{m\}_k$, where m is atomic, define the set

$$\mathsf{C}(x) = \{y \in \mathsf{sf} \mid \mathit{context}(y) = \mathit{context}(x), \mathit{left}(y) \neq \mathit{left}(x), \mathit{right}(y) = \mathit{right}(x), \mathit{op}(y) = \mathit{op}(x)\}$$

The algorithm to compute $\mathit{deriv}'(X, \Phi)$ is given in Algorithm 2. For each $x \in \mathsf{sf}$, we keep track of its status, which we denote by $\mathit{status}(x)$. It takes one of the values *raw*, *pending*, or *processed*. We also use a queue Q of formulas, with the corresponding *enqueue* and *dequeue* functions. It is easy to argue that whenever $\mathit{status}(x)$ becomes *pending*, it is the case that $x \in \mathit{deriv}'(X, \Phi)$. It is also the case that all *pending* formulas become *processed* once the algorithm terminates (if it does). Further, one can argue by induction on the size of proofs that for every formula $x \in \mathit{deriv}'(X, \Phi)$, eventually $\mathit{status}(x)$ becomes *pending*.

One can argue that the algorithm terminates in $O(N)$ time as follows. Each element enters Q at most once (when it is *raw*, and it becomes *pending* just before entering Q). We process each element of Q exactly once and mark it *processed* and dequeue it from the queue. For each element u of the queue, we spending constant time setting the status of $\mathit{left}(u)$ and $\mathit{right}(u)$ and perhaps enqueuing them. The sets $\mathsf{C}(u)$, $\mathsf{A}_\ell(u)$, $\mathsf{A}_r(u)$ and $\mathsf{O}_c(u)$ have no bound on their size, but for distinct u and u', the sets $\mathsf{A}_\ell(u)$ and $\mathsf{A}_\ell(u')$ are disjoint, and similarly for A_r. For distinct u and u', the sets $\mathsf{O}_c(u)$ and $\mathsf{O}_c(u')$ can have one element in common, namely $u \vee u'$. However, $u \vee u'$ will be marked *pending* the first time either u or u' is seen, and not be considered again. So across all elements in sf, the total time consumed in each of the **for all** blocks inside the **while** loop is $O(N)$. This gives us a linear time algorithm to compute $\mathit{deriv}'(X, \Phi)$.

Algorithm 2 Linear time algorithm for $deriv'(X, \Phi)$

```
$Q \leftarrow \emptyset$;
for all $x \in \Phi$ do
    if $op(x) = e$ then                                              ▷ $x$ is of the form $t = t'$
        $status(x) \leftarrow processed$;
    else if $op(x) = \prec$ and $left(x) = right(x)$ then               ▷ $x$ is of the form $m \prec m$
        $status(x) \leftarrow processed$;
    else                                                             ▷ $x$ is not of these two types
        $status(x) \leftarrow pending$;    $enqueue(Q, x)$;
    end if
end for
for all $x \in \mathsf{sf} \setminus \Phi$ : $status(x) \leftarrow raw$;
while $Q \neq \emptyset$ do
    $u \leftarrow dequeue(Q)$;
    if $op(u) = \wedge$ then
        for all $v \in child(u)$ such that $status(v) = raw$ do
            $status(v) \leftarrow pending$;    $enqueue(Q, v)$;          ▷ $u$ is the premise of the $\wedge e$ rule.
        end for
    end if
    for all $v \in \mathsf{A}_\ell(u)$ such that $status(right(v)) \neq raw$ and $status(v) = raw$ do
        $status(v) \leftarrow pending$;    $enqueue(Q, v)$;            ▷ $u$ is the left premise of the $\wedge i$ rule.
    end for
    for all $v \in \mathsf{A}_r(u)$ such that $status(left(v)) \neq raw$ and $status(v) = raw$ do
        $status(v) \leftarrow pending$;    $enqueue(Q, v)$;            ▷ $u$ is the right premise of the $\wedge i$ rule.
    end for
    for all $v \in \mathsf{O}_c(u)$ such that $status(v) = raw$ do
        $status(v) \leftarrow pending$;    $enqueue(Q, v)$;            ▷ $u$ is a premise of the $\vee i$ rule.
    end for
    for all $v \in \mathsf{C}(u)$ do
        if $status(v) \neq raw$ then
            for all $x \in \mathsf{sf}$ do
                $status(x) \leftarrow pending$;          ▷ $\perp$ rule allows us to derive any formula in $\mathsf{sf}$
            end for
        end if
    end for
    $status(u) \leftarrow processed$;
end while
$deriv'(X, \Phi) = \{u \in \mathsf{sf} \mid status(u) = processed\}$;
```

4 The active intruder

So far, we have spoken only of the term derivability problem, and thus implicitly, only the passive intruder case of security protocol verification. However, what we wish to verify are runs of protocols involving multi-sessions in the presence of an active intruder who has access to all communications on the network.

In general, security verification for protocols with unboundedly many concurrent sessions is undecidable [14], and interesting decidable subclasses have been identified [18, 7]. For practical purposes, it is customary to consider protocol runs involving only boundedly many multi-sessions, and while security verification is decidable, it is harder than term derivability: NP-complete, as opposed to polynomial time [19]. In the same spirit, we consider bounded protocols, but considering that the complexity of term derivability is high, we show that the active intruder case is no harder.

Protocols are typically specified as sequences of communications, but in formal analysis of protocols, it is convenient to consider a protocol 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. The set M denotes the set of nonces used in t that, in the context of a protocol run, 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 relaying 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 *non-modal* 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 A : *confirm* α and A : *deny* α.

A *knowledge state* ks is of the form $((X_A, \Phi_A)_{A \in Ag}, SD)$. 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, and $\Phi_A \subseteq \mathscr{A}$ is similarly the set of assertions accumulated. $SD \subseteq \{\{\alpha\}_{sd(B)} \mid B \in Ag, \alpha \in \mathscr{A}\}$ is the set of signed assertions transmitted thus far in the run. The *initial knowledge state* of a protocol Pr is $((X_A, \varnothing)_{A \in Ag}, \varnothing)$ where, for each A, $X_A = const(Pr) \cup Ag \cup \{sk(A)\} \cup \{pk(B), k(A, B) \mid B \in Ag\}$.

Let $ks = ((X_A, \Phi_A)_{A \in Ag}, SD)$ 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 $ks \xrightarrow{a} ks'$.

- $a = A!B : [(M)t, \{\alpha\}_{sd(A)}]$

- a is enabled at ks iff
    1. $M \cap X_C = \emptyset$ for all C,
    2. $X_A \cup M \vdash_{dy} t$, and
    3. $X_A \cup M, \Phi_A \vdash_{al} \alpha$.
- $ks \xrightarrow{a} ks'$ iff
    1. $X'_A = X_A \cup M$, $X'_I = X_I \cup \{t\}$,
    2. $\Phi'_I = \Phi_I \cup \{\alpha, A\ says\ \alpha\}$, and
    3. $SD' = SD \cup \{\{\alpha\}_{sd(A)}\}$.

- $a = A?B{:}[t, \{\alpha\}_{sd(B)}]$
  - a is enabled at ks iff
    1. $\{\alpha\}_{sd(B)}$ is verified as signed by B,
    2. $X_I \vdash_{dy} t$, and
    3. either $X_I, \Phi_I \vdash_{al} \alpha$ or $\{\alpha\}_{sd(B)} \in SD$.
  - $ks \xrightarrow{a} ks'$ iff
    1. $X'_A = X_A \cup \{t\}$ and
    2. $\Phi'_A = \Phi_A \cup \{B\ says\ \alpha\}$.

- $a = A : confirm\ \alpha$
  - a is enabled at ks iff $X_A, \Phi_A \vdash_{al} \alpha$.
  - $ks \xrightarrow{a} ks'$ iff $ks = ks'$.

- $a = A : deny\ \alpha$
  - a is enabled at ks iff $X_A, \Phi_A \nvdash_{al} \alpha$.
  - $ks \xrightarrow{a} ks'$ 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 $A\ says\ \alpha$). A recipient agent need not verify the assertion because of our perfect assertion assumption, by which the system allows only true assertions to be passed on.

In a concrete realization of our model, this situation may be implemented by a trusted third party verifier (TTP) who 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 as such 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.

The notions of substitutions and role instances are defined in the standard manner. An *role instance* of a protocol Pr is a tuple $ri = (\eta, \sigma, lp)$, where η is a role of Pr, σ is a substitution 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 $TS(Pr)$.

Definition 24 (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 25

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 K disjunctions, for a fixed K.*

Proof

1. A run of Pr starting from a 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 K disjunctions for a fixed K, along each run we consider, there will be at most $k * K$ 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. ⊣

5 Example protocols and attacks

5.1 Protocol 1:

An agent A sends two encrypted votes $\{v_1\}_{k_A}$ and $\{v_2\}_{k_A}$ to the TTP T, who knows the key k_A. Each vote is for a separate 'position', and A claims that the person picked is one of two possible candidates. T sends c_A if A's assertions about the votes are confirmed, 0 otherwise. Disjunction allows A to convey partial information about the votes. v_1 is secret to A and T.

- $A \to T : \{v_1\}_{k_A},\ \{(a \prec \{v_1\}_k) \vee (b \prec \{v_1\}_k)\}$
- $A \to T : \{v_2\}_{k_A},\ \{(c \prec \{v_2\}_k) \vee (d \prec \{v_2\}_k)\}$
- If T confirms these two assertions, $T \to A : c_A$. Otherwise, $T \to A : 0$.

An attack can be found via disjunction elimination in the case where $c = a$, say and an agent votes for the same candidate a for both positions. Here, A sends the same term $\{a\}_k$ twice, and two assertions of the form $(a \prec \{a\}_k) \vee (b \prec \{a\}_k)$ and $(a \prec \{a\}_k) \vee (d \prec \{a\}_k)$ about it. This allows the intruder to know that the actual vote is a.

5.2 Protocol 2:

A and B generate a vote, which they want principals C and D to agree to, and then send to the trusted third party T. However, they do not want these parties 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 $t = \{v_T\}_k$.

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

We now demonstrate an attack. Suppose there is a session S_1 where a and b take values a_1 and b_1, where C 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 C's message to D in S_2 from S_1, although C might not wish to agree in S_2. Moreover, the intruder might also find the actual vote by or-elimination.

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. Hrițcu 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] B. Blanchet and A. Podelski. Verification of Cryptographic Protocols: Tagging Enforces Termination. In *Proceedings of FoSSaCS'03*, LNCS 2620, pages 136–152, 2003.

[8] M. Burrows, M. Abadi and R. M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, Feb 1990.

[9] J. Benaloh. Cryptographic capsules: A disjunctive primitive for interactive protocols. In *Proceedings of CRYPTO '86*, LNCS 263, pages 213–222, 1987.

[10] 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.

[11] 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.

[12] 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.

[13] D. Dolev and A. Yao. On the Security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[14] N. A. Durgin, P. D. Lincoln, J. C. Mitchell and A. Scedrov. Multiset Rewriting and the Complexity of Bounded Security Protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[15] 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.

[16] Y. Gurevich and I. Neeman. Infon logic: the propositional case. *ACM Transactions on Computational Logic*, 12(2):9:1–9:28, 2011.

[17] 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.

[18] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proceedings of FSTTCS 2003*, LNCS 2914, 363–374, 2003.

[19] 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.