Tagging Makes Secrecy Decidable With Unbounded Nonces As Well *

R. Ramanujam and S. P. Suresh

The Institute of Mathematical Sciences
C.I.T. Campus, Chennai 600 113, India.
E-mail: {jam,spsuresh}@imsc.res.in

Abstract. Tagging schemes have been used in security protocols to ensure that the analysis of such protocols can work with messages of bounded length. When the set of nonces is bounded, this leads to decidability of secrecy. In this paper, we show that tagging schemes can be used to obtain decidability of secrecy even in the presence of unboundedly many nonces.

1 Background

Security protocols are specifications of communication patterns which are intended to let agents share secrets over a public network. They are required to perform correctly even in the presence of malicious intruders who listen to the message exchanges that happen over the network and also manipulate the system (by blocking or forging messages, for instance). An obvious correctness requirement is that of secrecy: an intruder cannot read the contents of a message intended for others.

The presence of intruders necessitates the use of encrypted communication. It has been widely acknowledged that even if perfect cryptographic tools are used, desired security goals may not be met, due to logical flaws in the design of protocols. Thus *automatic verification of security protocols* is an important and worthwhile enterprise. This is complicated by the fact that security protocols are in general infinite state systems. As such, it is to be expected that it is not possible to verify even simple properties like secrecy of such systems. It has been formally proved in ([7], [9], [1]) that in fact, the secrecy problem is undecidable. The prominent sources of undecidability are unbounded message length and unbounded number of nonces.

The undecidability results seem to be at variance with the high degree of success achieved in verifying not just secrecy but also other more complicated properties of security protocols in practice. Hence there have been many attempts to prove decidability by imposing reasonable restrictions on the model. [7] shows that when both message length and the number of nonces is bounded, the secrecy problem is DEXPTIME-complete. [11] and [16] essentially place bounds on the

* We thank the anonymous referees for detailed comments, which have helped to improve the presentation greatly, both in terms of the modelling and the proofs.

number of sessions that can occur in any run of the protocol, thereby obtaining decidability. [10] proves decidability for a syntactic subclass and our work is closest in spirit to this work.

In earlier work, we separately studied the secrecy problem in the setting of bounded-length messages ([13]) and in the setting of boundedly many nonces ([14]), showing decidability fo subclasses of protocols in both cases. In this paper, we prove decidability for the subclass of tagged protocols without assuming any external bounds. The tagging scheme ensures primarily that no two encrypted subterms of distinct communications in the protocol specification are unifiable. Similar schemes have been used in [2] to prove the termination of their verification algorithm, and in [8] to prevent type-flaw attacks.

Our decidability proof works by first tackling the problem of unbounded message length and then the problem of unboundedly many nonces. Message length can get unbounded when the intruder substitutes nonatomic terms for atomic terms. We show that our tagging scheme ensures that the honest agents do not make a criticial use of such terms for learning new information, and thus it suffices for verification of secrecy to consider only well-typed runs where nonces are instantiated only with atomic data. We next show that whenever a run of a tagged protocol has a send action a such that none of the succeeding receive actions has encrypted terms in common with a, then we can eliminate a from the run and perform a systematic renaming of the nonces to arrive at a run of shorter length which is leaky iff the original run is. We also prove that repeating this process yields us a run of bounded length, and show that it suffices for decidability. A proof outline is provided in Section 3, the details of which can be found in [15].

The technique used here should be contrasted with approaches which impose restrictions on the use of the tupling operator ([1], [6]), or use more stringent admissibility criteria like [4] which uses techniques from tree automata theory to show decidability for the class of protocols in which every agent copies at most one piece of any message it receives into any message it sends, or approaches like [3], where an abstraction of nonces is used to prove the termination of a verification algorithm for a large class of protocols. Apart from decidability, the model presented here has other interesting features like send-admissibility, which formalises a notion of reasonableness for protocols, and synth and analz proofs, which formalise how messages are generated and received terms are analyzed.

2 Security Protocol Modelling

In this section we briefly present our model of security protocols. Our modelling is close to the inductive approach of [12] in many respects. A more detailed presentation can be found in [15].

Actions

We assume a finite set of *agents* Ag with a special *intruder* $I \in Ag$. The set of *honest agents*, denoted Ho, is defined to be $Ag \setminus \{I\}$. We assume an infinite set of *nonces* N. The set of keys K is given by $K_0 \cup K_1$ where K_0 is an infinite set and $K_1 = \{k_{AB}, privk_A, pubk_A \mid A, B \in Ag, A \neq B\}$. $pubk_A$ is A's *public key* and $privk_A$ is its *private key*. k_{AB} is the *(long-term) shared key* of A and B. For each $A \in Ag$, $K_A \stackrel{\text{def}}{=} \{k_{AB}, k_{BA}, pubk_A, privk_A, pubk_B \mid B \in Ag, B \neq A\}$. For $k \in K$, $\overline{k}$, the *inverse key* of k, is defined as follows: $\overline{pubk_A} = privk_A$ and $\overline{privk_A} = pubk_A$ for all $A \in Ag$, and $\overline{k} = k$ for all the other keys. The set of *basic terms* $\mathcal{T}_0$ is defined to be $K \cup N \cup Ag$.

The set of *information terms* is defined to be

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

where m ranges over $\mathcal{T}_0$ and k ranges over K.

The notion of subterm of a term is the standard one — $ST(m) = \{m\}$ for $m \in \mathcal{T}_0$; $ST((t_1, t_2)) = (t_1, t_2) \cup ST(t_1) \cup ST(t_2)$; and $ST(\{t\}_k) = \{\{t\}_k\} \cup ST(t) \cup ST(k)$. t' is an *encrypted subterm* of t if $t' \in ST(t)$ and t' is of the form $\{t''\}_k$. $EST(t)$ denotes the set of encrypted subterms of t.

An *action* is either a *send action* of the form $A!B{:}(M)t$ or a *receive action* of the form $A?B{:}t$ where: $A \in Ho, B \in Ag$ and $A \neq B$; $t \in \mathcal{T}$; and $M \subseteq ST(t) \cap N$. In a send action of the form $A!B{:}(M)t$, M is the set of nonces freshly generated by A just before sending t. For simplicity of notation, we write $A!B{:}t$ instead of $A!B{:}(\emptyset)\ t$. The set of all actions is denoted by Ac.

Note that we do not have explicit intruder actions in the model. As will be clear from the definition of updates caused by actions, every send action is implicitly considered to be an instantaneous receive by the intruder, and similarly, every receive action is considered to be an instantaneous send by the intruder. Thus the agent B is (merely) the intended receiver in $A!B{:}(M)t$ and the purported sender in $A?B{:}t$.

For a of the form $A!B{:}(M)t$, $term(a) \stackrel{\text{def}}{=} t$ and $NT(a) \stackrel{\text{def}}{=} M$. For a of the form $A?B{:}t$, $term(a) \stackrel{\text{def}}{=} t$ and $NT(a) \stackrel{\text{def}}{=} \emptyset$. $NT(a)$ stands for *new terms* generated during action a. The notation is appropriately extended so that we can talk of $terms(\eta)$ and $NT(\eta)$ for $\eta \in Ac^*$. $ST(a)$ and $EST(a)$ have the obvious meanings, $ST(term(a))$ and $EST(term(a))$ respectively. $\eta\lceil A$, A's view of η, is the subsequence of η obtained by projecting it down to the set of A-actions.

Protocols

Definition 2.1 *An* information state *s is a tuple $(s_A)_{A \in Ag}$ where $s_A \subseteq \mathcal{T}$ for each agent A. $\mathcal{S}$ denotes the set of all information states. For a state s, we define $ST(s)$ to be $\bigcup_{A \in Ag} ST(s_A)$.*

Definition 2.2 *A* **protocol** *is a pair $\mathsf{Pr} = (\mathsf{C}, \eta)$ where C, the set of* constants *of Pr, denoted $\mathsf{CT}(\mathsf{Pr})$, is a subset of $\mathcal{T}_0$, and $\eta \in Ac^+$ is the* body *of Pr.*

Given a protocol $\mathsf{Pr} = (\mathsf{C}, \eta)$, $Roles(\mathsf{Pr})$, the set of roles *of Pr, is defined to be the set $\{\eta\lceil A \mid A \in Ag$ and $\eta\lceil A \neq \varepsilon\}$.*

In the literature, protocols are informally specified as a sequence of communications of the form $A \to B{:}t$. Such protocols can be presented in the above formalism by splitting each communication into a send action and a matching receive action. Protocols which are presented as a finite set of roles can also be presented in the above formalism.

Definition 2.3 *Given a protocol* Pr*, we define the* initial state of Pr*, denoted* $s_0(\mathsf{Pr})$*, to be* $(T_A)_{A \in Ag}$ *where for all* $A \in Ho$*,* $T_A = \mathsf{CT}(\mathsf{Pr}) \cup K_A$ *and* $T_I = \mathsf{CT}(\mathsf{Pr}) \cup K_I \cup \{z\}$, where z is a fixed nonce which is assumed to be different from all the nonces in $\mathsf{CT}(\mathsf{Pr})$.

As we have mentioned earlier, we do not explicitly model intruder actions. Thus we do not explicitly model the phenomenon of the intruder generating new nonces in the course of a run, as is done in some other models (for instance, [7]). An alternative would be to provide an arbitrary set of nonces and keys to the intruder in the initial state. We follow the approach of just providing the intruder with the fixed nonce z in the initial state. It is a symbolic name for the set of new data the intruder might generate in the course of a run. This suffices for the analysis we perform in our proofs later. We will ensure as we develop the model that z is not generated as a fresh nonce by any honest agent in the course of a run of Pr.

A *substitution* σ is a map which maps nonces to arbitrary terms, keys to keys and agent names to agent names. Substitutions are extended to terms, actions, and sequences of actions in a straightforward manner. A substitution σ is said to be *well-typed* iff for $n \in N$, $\sigma(n) \in N$. A substitution σ is said to be *suitable for an action* a iff it maps each distinct nonce (or key) in $NT(a)$ to a distinct nonce (or key, as the case may be), and has disjoint ranges for $NT(a)$ and $ST(a) \setminus NT(a)$. σ is suitable for $a_1 \cdots a_\ell$ iff for all $i \leq \ell$, σ is suitable for a_i. σ is said to be suitable for a protocol Pr if $\sigma(t) = t$ for all constants $t \in \mathsf{CT}(\mathsf{Pr})$.

Given a protocol Pr, a triple (η, σ, lp) is an *event* of Pr iff $\eta \in Roles(\mathsf{Pr})$, σ is a substitution suitable for Pr and η, and $1 \leq lp \leq |\eta|$. $Events(\mathsf{Pr})$ is the set of all events of Pr. An event (η, σ, lp) of Pr is said to be well-typed iff σ is well-typed. For an event $e = (\eta, \sigma, lp)$ of Pr with $\eta = a_1 \cdots a_\ell$, $act(e) \stackrel{\text{def}}{=} \sigma(a_{lp})$. If $lp < |\eta|$ then $(\eta, \sigma, lp) \rightarrow_\ell (\eta, \sigma, lp + 1)$. For two events e and e' of Pr, $e' \in LP(e)$, the *local past* of e, iff $e' \stackrel{+}{\rightarrow}_\ell e$. For any event e of Pr, $NT(e)$ will be used to denote $NT(act(e))$ and similarly for $term(e)$, $ST(e)$, $EST(e)$, etc.

Runs Of Protocols

Definition 2.4 *A* sequent *is of the form* $T \vdash t$ *where* $T \subseteq \mathcal{T}$ *and* $t \in \mathcal{T}$.

An analz*-proof (*synth*-proof)* π *of* $T \vdash t$ *is an inverted tree whose nodes are labelled by sequents and connected by one of the* analz*-rules (*synth*-rules) in Figure 1, whose root is labelled* $T \vdash t$*, and whose leaves are labelled by instances of the* Ax_a *rule (*Ax_s *rule). For a set of terms* T*,* $\mathsf{analz}(T)$ *(*$\mathsf{synth}(T)$*) is the set of terms* t *such that there is an* analz*-proof (*synth*-proof) of* $T \vdash t$*.*

For ease of notation, $\mathsf{synth}(\mathsf{analz}(T))$ *is denoted by* $\overline{T}$*.*

Definition 2.5 *The notions of an action enabled at a state and update of a state on an action are defined as follows:*

$$\frac{}{T \cup \{t\} \vdash t}\,\mathsf{Ax}_a \qquad \frac{T \vdash (t_1, t_2)}{T \vdash t_i}\,\mathsf{split}_i \qquad \frac{T \vdash \{t\}_k \quad T \vdash \overline{k}}{T \vdash t}\,\mathsf{decrypt} \qquad \frac{T \vdash \{\{t\}_k\}_{\overline{k}}}{T \vdash t}\,\mathsf{reduce}$$

analz-rules

$$\frac{}{T \cup \{t\} \vdash t}\,\mathsf{Ax}_s \qquad \frac{T \vdash t_1 \quad T \vdash t_2}{T \vdash (t_1, t_2)}\,\mathsf{pair} \qquad \frac{T \vdash t \quad T \vdash k}{T \vdash \{t\}_k}\,\mathsf{encrypt}$$

synth-rules

Fig. 1. analz and synth rules.

- $A!B{:}(M)t$ *is* enabled *at* s *iff* $t \in \overline{s_A \cup M}$, *and* $M \cap ST(s) = \emptyset$.
- $A?B{:}t$ *is* enabled *at* s *iff* $t \in \overline{s_I}$.
- $update(s, A!B{:}(M)t) \stackrel{\text{def}}{=} s'$ *where* $s'_A = s_A \cup M \cup \{t\}$, $s'_I = s_I \cup \{t\}$, *and for all* $C \in Ag \setminus \{A, I\}$, $s'_C = s_C$.
- $update(s, A?B{:}t) \stackrel{\text{def}}{=} s'$ *where* $s'_A = s_A \cup \{t\}$ *and for all* $C \in Ag \setminus \{A\}$, $s'_C = s_C$.

$update(s, \varepsilon) = s$, $update(s, \eta \cdot a) = update(update(s, \eta), a)$.

Given a protocol Pr, and a sequence $\xi = e_1 \cdots e_k$ of events of Pr, $infstate(\xi)$ is defined to be $update(s_0(\mathsf{Pr}), act(e_1) \cdots act(e_k))$. We say that an event e of Pr is *enabled at* ξ iff $LP(e) \subseteq \{e_1, \cdots, e_k\}$ and e is enabled at $infstate(\xi)$.

Definition 2.6 *Given a protocol* Pr*, the set of* runs *of* Pr*, denoted by* $\mathcal{R}(\mathsf{Pr})$*, is defined to be the set of all sequences* $e_1 \cdots e_k$ *of events of* Pr *such that for all* $i : 1 \leq i \leq k$, e_i *is enabled at* $e_1 \cdots e_{i-1}$*. A run is said to be* well-typed *iff every event occurring in it is well-typed.*

Well-formed Protocols

$(A!B{:}(M)t, C?D{:}t')$ is said to be a *matching send-receive pair* iff $A = D$, $B = C$, and $t = t'$. Note that we require *syntactic equality* of the terms t and t' rather than just unifiability.

Given a protocol Pr, a sequence of actions $\eta = a_1 \cdots a_\ell$ is said to be *send-admissible* with respect to Pr iff for all $i \leq \ell$, if a_i is a send action then a_i is enabled at $update(s_0(\mathsf{Pr}), a_1 \cdots a_{i-1})$.

Definition 2.7 *A* well-formed protocol *is a protocol* $\mathsf{Pr} = (\mathsf{C}, a_1 b_1 \cdots a_\ell b_\ell)$ *where* (a_i, b_i) *is a matching send-receive pair for all* $i : 1 \leq i \leq \ell$ *and* η *is send-admissible with respect to* Pr.

Well-formed protocols formalise a notion of "reasonableness" of protocol specifications. Almost all the standard protocols studied in the literature are well-formed. The following useful fact follows easily from the definition of well-formed protocols and from some basic properties of the synth and analz operators.

Proposition 2.8 *Suppose that* $\mathsf{Pr} = (\mathsf{C}, \eta)$ *is a well-formed protocol. Then for all roles* ζ *of* Pr *and* σ *suitable for* ζ *and* Pr, ζ *and* $\sigma(\zeta)$ *are send-admissible with respect to* Pr.

Tagged Protocols

While well-formed protocols enforce a reasonableness condition at the level of protocol specifications, we must note that they still allow for quite unreasonable behaviours. Substituting encrypted terms for nonces can give the intruder the ability to circumvent the protocol. For instance, a communication of the form $A \to B : \{(A, \{x\}_B)\}_B$ in the protocol allows the intruder to capture it and send it on to B as: $I \to B : \{(I, \{\{(A, \{x\}_B)\}_B\}_B)\}_B$. This goes against the reasonable requirement that B expects only terms of encryption depth 2 whereas here B gets a term of depth 3. We thus look for mechanisms that enforce only "reasonable runs". *Tagging* is one such mechanism that seeks to distinguish between terms of different encryption depth as above. More specifically, tags are just constants which act as message identifiers and are attached to some of the encrypted subterms of messages which are communicated during a run. The use of tags has the effect of preventing the intruder from passing off a term $\sigma(\{t\}_k)$ as $\sigma'(\{t'\}_{k'})$ in some run of a protocol while $\{t\}_k$ and $\{t'\}_{k'}$ are intended to be distinct terms in the protocol specification. We also use tagging to associate every receive action occurring in a run with its *corresponding send* (if there exists one).

Definition 2.9 *A well-formed protocol* $\mathsf{Pr} = (\mathsf{C}, \eta)$ *with* $\eta = a_1 b_1 \cdots a_\ell b_\ell$ *is called a* tagged protocol *iff: for all* $t \in EST(\eta)$ *there exists* $\mathsf{c}_t \in \mathsf{C}$*, and for all* $i \leq \ell$ *there exists* $n_i \in NT(a_i)$ *such that:*

- *for all* $i, j \leq \ell$, $t \in EST(a_i)$*, and* $t' \in EST(a_j)$: *if* $\mathsf{c}_t = \mathsf{c}_{t'}$ *then* $t = t'$ *and* $i = j$*, and*
- *for all* $i \leq \ell$ *and all* $t \in EST(a_i)$, $t = \{(\mathsf{c}_t, (n_i, u))\}_k$ *for some* u *and* k.

Most of the standard protocols occurring in the literature (see [5] for example) can be easily tagged to obtain "equivalent protocols", such that for any run ξ of the original protocol which involves only honest agents, the tagged version of ξ is a run of the transformed protocol, and for all runs ξ of the transformed protocol, the untagged version of ξ is a run of the original protocol. (Thus the transformation does not limit the honest agents' capabilities while at the same time not introducing more attacks). The protocols for which this transformation cannot be effected are those which contain "blind copies" like the Woo-Lam protocol Π (as presented in [5]). It is to be noted that the schemes presented in [8] and [2] — with reference to Definition 2.9, these are equivalent to using just the c_t tags to distinguish between distinct terms in $EST(\eta)$ — work even for protocols with "blind copies", in fact those schemes work for all well-formed protocols. An important point worth noting here is that including the tags in the protocol specification stage rather than later, in the run generation stage, means that the reasonableness of runs is enforced by checks performed by the honest participants of the protocol.

Tagging each send action with a *new nonce* might seem a costly operation, since it is nontrivial to keep generating many *distinct*, *unguessable* random numbers. But the proofs (in particular, the proof of item 2 of Proposition 2.10, which is the only place where this property of tagged protocols is used) only require the fact that the n_i's are instantiated with distinct values for distinct substitutions, and not the fact that they are unguessable values. Thus the n_i's are playing the role of *sequence numbers*, and are as such easy to implement.

The following property, useful for proofs later, can be easily seen to be a direct consequence of the definition of tagged protocols.

Proposition 2.10 *Suppose* $\mathsf{Pr} = (\mathsf{C}, a_1 b_1 \cdots a_\ell b_\ell)$ *is a tagged protocol. Then the following statements hold:*

- *For all* σ, σ' *suitable for* Pr *and for all* $i, j \leq \ell$, $t \in EST(a_i)$, $t' \in EST(a_j)$, *if* $\sigma(t) = \sigma'(t')$ *then* $t = t'$ *and* $i = j$.
- *Suppose* $e_1 \cdots e_k$ *is a well-typed run of* Pr. *For all receive events* $e_j (j \leq k)$, *there is at most one send event* e_i *such that* $EST(e_i) \cap EST(e_j) \neq \emptyset$.

The Secrecy Problem

Definition 2.11 *A basic term* $m \in \mathcal{T}_0$ *is said to be* secret at state s *iff there exists* $A \in Ho$ *such that* $m \in \mathsf{analz}(s_A) \setminus \mathsf{analz}(s_I)$. *Given a protocol* Pr *and* $\xi \in \mathcal{R}(\mathsf{Pr})$, m *is said to be secret at* ξ *if it is secret at infstate*(ξ). ξ *is* leaky *iff there exists a basic term* m *and a prefix* ξ' *of* ξ *such that* m *is secret at* ξ' *and not secret at* ξ. *The secrecy problem is the problem of determining for a given protocol* Pr *whether some run of* Pr *is leaky.*

Thus we say that a run is leaky if some atomic term is secret at some intermediate state of the run but is revealed to the intruder at the end of the run. It is possible that there are protocols for which leaks of the above form do not constitute a breach of security. A more general notion would be to allow the user to specify certain secrets which should not be leaked and check for such leaks. We believe that the techniques we use here can be adapted to prove the decidability of the more general problem as well.

While the general secrecy problem has been proved to be undecidable in various settings ([7], [9]), the main result of this paper is the following decidability result.

Theorem 2.12 *The secrecy problem for tagged protocols is decidable.*

The theorem follows from a series of observations, which will be proved in the next section.

1. If a tagged protocol has a leaky run, then it has a well-typed leaky run.

2. If a tagged protocol has a well-typed leaky run, then it has a *good* well-typed leaky run.
3. All good well-typed runs are of bounded length.

Properties Of synth And analz

We now state some basic properties of the synth and analz operators. The proofs are by a routine induction on proof trees.

Proposition 2.13 *Let $T, T' \subseteq \mathcal{T}$ and σ be a substitution. Then the following properties hold:*

$$T \subseteq \mathsf{analz}(T) \text{ and } T \subseteq \mathsf{synth}(T).$$
$$\text{if } T \subseteq T' \text{ then } \mathsf{analz}(T) \subseteq \mathsf{analz}(T') \text{ and } \mathsf{synth}(T) \subseteq \mathsf{synth}(T').$$
$$\mathsf{analz}(\mathsf{analz}(T)) = \mathsf{analz}(T) \text{ and } \mathsf{synth}(\mathsf{synth}(T)) = \mathsf{synth}(T).$$
$$\overline{\overline{T}} = \mathsf{analz}(\overline{T}) = \overline{T}.$$
$$\sigma(\mathsf{analz}(T)) \subseteq \mathsf{analz}(\sigma(T)) \text{ and } \sigma(\mathsf{synth}(T)) \subseteq \mathsf{synth}(\sigma(T)).$$

Definition 2.14 *t is a* minimal term *of T if $t \in T$ and $t \notin \mathsf{synth}(T \setminus \{t\})$. $\mathsf{min}(T)$ denotes the set of minimal terms of T.*

Suppose t is not a minimal term of T. Then from the definition it follows that $\mathsf{synth}(T) = \mathsf{synth}(T \setminus \{t\})$. Since T is generally used as a representative for $\mathsf{synth}(T)$, if T contains a nonminimal term then there is a smaller representative for $\mathsf{synth}(T)$. Thus nonminimal terms can be viewed as redundant in such situations.

Proposition 2.15 *For any set of terms T, $T \subseteq \mathsf{synth}(\mathsf{min}(T))$, $\mathsf{synth}(T) = \mathsf{synth}(\mathsf{min}(T))$ and $\overline{T} = \mathsf{synth}(\mathsf{min}(\mathsf{analz}(T)))$.*

3 Decidability

Reduction to well-typed runs: We outline the proof of Theorem 2.12. The first step is to prove that for all runs of a tagged protocol there is an equivalent well-typed run which preserves leakiness. Towards this, we define, for any substitution σ, σ_z as follows: for all $x \in \mathcal{T}_0$, (if $x \in N$ and $\sigma(x) \notin N$ then $\sigma_z(x) = z$, otherwise $\sigma_z(x) = \sigma(x)$). Suppose Pr is a tagged protocol and $e_1 \cdots e_k$ is a run of Pr where each $e_i = (\eta_i, \sigma_i, lp_i)$. For every $i \leq k$, define $e'_i = (\eta_i, (\sigma_i)_z, lp_i)$. Note that e'_i is well-typed by definition. We prove the reduction to well-typed runs by showing that $e'_1 \cdots e'_k$ is a run of Pr which is leaky iff $e_1 \cdots e_k$ is. It is easy to see that the above transformation (of replacing the σ_i's by $(\sigma_i)_z$'s) does not affect send-admissibility, and hence all send events e'_i are enabled at $e'_1 \cdots e'_{i-1}$. For a receive event e'_i, we know that $t_i \in \overline{T_{i-1}}$ (where $t_i = term(e_i)$ and $T_{i-1} = (infstate(e_1 \cdots e_{i-1}))_I$). If we show that $t'_i \in \overline{T'_{i-1}}$, it would follow that $e'_1 \cdots e'_k$ is a run of Pr, and also that it is leaky iff $e_1 \cdots e_k$ is (since replacing the σ_i's with $(\sigma_i)_z$'s doesn't affect new terms generated during the actions, and since the set of basic terms known to the intruder at the corresponding states in both the runs is the same).

To prove that $t'_i \in \overline{T'_{i-1}}$ we show how to transform the proof that $t_i \in \overline{T_{i-1}} = \mathsf{synth}(\mathsf{min}(\mathsf{analz}(T)))$. This consists of a synth-proof π of $\mathsf{min}(\mathsf{analz}(T_{i-1})) \vdash t_i$, and an analz-proof ϖ_t of $T_{i-1} \vdash t$ for each t labelling a leaf of π. Note that every term t occurring in a leaf of π is either a nonce or an encrypted term (since tuples can be synthesiized from their components and hence are not in $\mathsf{min}(\mathsf{analz}(T_{i-1}))$). Every u labelling a node of π or one of the ϖ_t's is a subterm of t_j for some $j \leq i$. Letting $r_i = \eta_i(lp_i)$ for each i, we see that t_i corresponds to $\sigma_i(r_i)$. Suppose we type the root of π with (σ_i, r_i). This will induce a partial typing of π with types of the form (σ_i, w) where $w \in ST(r_i)$. Suppose for all leaves of π typed (σ_i, w) where $w \in EST(r_i)$ it is shown that $(\sigma_i)_z(w) \in \mathsf{analz}(T'_{i-1})$. Then it can be easily shown that $t'_i = (\sigma_i)_z(r_i) \in \overline{T'_{i-1}}$. (Some of the non-leaf nodes of π might be typed (σ_i, m) for some nonce m. In such cases it should be noted that $(\sigma_i)_z(m) = z \in T_0 = T'_0$.)

We now consider encrypted terms t occurring in the leaves of π which have a type (σ_i, w) with $w \in EST(r_i)$. If $t = \sigma_j(w')$ for some $j < i$ and $w' \in EST(r_j)$ then the $\mathsf{tagging\ scheme}$ (specifically, item 1 of Proposition 2.10) ensures that $w = w'$. So if we prove that $(\sigma_j)_z(w') \in \mathsf{analz}(T'_{i-1})$ for some $j \leq i$ and some $w' \in ST(r_j)$ such that w' is an encrypted term when t is, then it would follow that $t'_i \in \overline{T'_{i-1}}$.

Now consider an analz-proof among the ϖ_t's. We can define the set of types for any node of such a proof by letting each leaf labelled by a term u be typed by $\{(\sigma_j, r_j) \mid j \leq i \text{ and } \sigma_j(r_j) = u\}$. This induces a set of types for each node. We say that a type (σ, r) matches a term t iff $\sigma(r) = t$ and r preserves the outermost structure of t, in particular r is an encrypted term when t is. ϖ_t is well-typed if the above induced typing on ϖ_t types its root with a type which matches t. We can prove that for any $t \in \mathsf{analz}(T_{i-1})$ there is a well-typed analz-proof of $T_{i-1} \vdash t$. This is proved by induction on i. The main technical issue here is to handle the case when a non-atomic term u occurs in a node and is typed only by nonces. By considering various cases we show that $u \in \overline{T_{i-2}}$, thus allowing us to handle this by the induction hypothesis. Once we have proved this, it is a straightforward induction on proofs to show that $(\sigma_j)_z(r) \in \mathsf{analz}(T'_{i-1})$. This concludes the reduction to well-typed runs.

Reduction to good runs: We now show that for detecting leaks it suffices to consider runs that satisfy a specific 'goodness' condition.

Definition 3.1 *Suppose* $\mathsf{Pr} = (\mathsf{C}, \eta)$ *is a tagged protocol and* $\xi = e_1 \cdots e_k$ *is a run of* Pr. *For* $i, j \leq k$, e_j *is called a* good successor *of* e_i *(and* e_i *a* good predecessor *of* e_j*) in* ξ *iff:* $i < j$ *and either* $e_i \to_\ell e_j$, *or* $EST(e_i) \cap EST(e_j) \neq \emptyset$.

For $i \leq k$, e_i *is called a* good event *in* ξ *iff either* $i = k$ *or there is some* $j > i$ *such that* e_j *is a good successor of* e_i. e_i *is called a* bad event *in* ξ *iff it is not a good event in* ξ. *A run* ξ *is called a* good run *iff all its events are good. A subsequence* $e_1 \cdots e_r$ *of* ξ *is called a* good path *in* ξ *iff for all* $j < r$, e_{j+1} *is a good successor of* e_j *in* ξ.

If e_j is a good successor of e_i then it is possible that e_j strongly depends on e_i in the sense that elimination of e_i from ξ disables e_j. If e_i is a bad predecessor of e_j then e_i can be eliminated while still enabling some "renamed variant" of e_j.

We now prove that whenever a tagged protocol Pr has a well-typed leaky run, it has a good well-typed leaky run. Fix a tagged protocol $\mathsf{Pr} = (\mathsf{C}, a_1 b_1 \cdots a_\ell b_\ell)$ and a leaky run $\xi = e_1 \cdots e_k$ of Pr such no proper prefix ξ_1 of ξ is leaky. If ξ is a good run, we are done; otherwise, there is a bad event occurring in ξ. Let r be the index of the latest bad event in ξ. Let $T = \mathcal{T}_0 \cap (\mathsf{analz}(T_r) \setminus \mathsf{analz}(T_{r-1}))$ (where $T_i = (\mathit{infstate}(e_1 \cdots e_i))_I$). Since $e_1 \cdots e_r$ is not leaky, there cannot be an $m \in T$ and $r' < r$ such that m is secret at $e_1 \cdots e_{r'}$. Thus $T \subseteq NT(e_r)$. Let τ be a substitution which maps every $m \in T$ to z and is identity otherwise. For all $e_i = (\eta_i, \sigma_i, lp_i)$ let $e'_i = (\eta_i, \tau \circ \sigma_i, lp_i)$ where $(\tau \circ \sigma_i)(t) = \tau(\sigma_i(t))$ for all t. We now show that $\xi' = e'_1 \cdots e'_{r-1} e'_{r+1} \cdots e'_k$ is a run of Pr, and that it is leaky; but the index of the latest bad event in it is less than r, and hence we can repeat the process, eventually obtaining a good run.

We first take up the task of proving that ξ' is a run of Pr. We first note that the bad event e_r is not in the local past of any other event and also the substitution τ does not affect the new terms generated by events other than e_r and hence does not affect send-admissibilty. Thus all the send events of ξ' are still enabled by the events occurring earlier. It is only the receive events we have to worry about. Here again if e_r is a receive event, then it is easy to see that $T = \emptyset$, i.e., nothing new is learnt by the intruder because of e_r, and hence enabledness of the other events is not affected even if e_r is eliminated from ξ. Thus again ξ' is a run of Pr. The nontrivial case is when e_r is a send event and e_q is a receive event for some $q > r$. In this case we know that $t_q \in \overline{T_{q-1}}$ (t_i denotes $\mathit{term}(e_i)$). If we show that $t_q \in \overline{(T_{q-1} \cup T) \setminus \{t_r\}}$ we are through, since $\tau(T) = \{z\} \subseteq T_0$ and hence $\tau(t_q) \in \overline{\tau(T_{q-1} \setminus \{t_r\})}$.

We first show that for all $q : r < q \leq k$ and all analz-proofs π whose root is labelled $T_q \vdash u$ and such that for all $T_q \vdash t$ labelling the non-root nodes of π, t is not secret at $e_1 \cdots e_{q-1}$, $u \in (\mathsf{analz}(T_r) \cap ST(t_r)) \cup \mathsf{analz}(T_q^{-r} \cup T)$. The intuition behind the proof is that if the proof has a $\mathsf{decrypt}$ node involving the terms $\{t\}_k$ and $\overline{k}$ then $\overline{k}$ itself is not secret at the point when $\{t\}_k$ is revealed to the intruder, and hence t is known to the intruder at the point when $\{t\}_k$ is known. Thus depending on whether t_r occurs in the leftmost leaf of π or not, $u \in \mathsf{analz}(T_r) \cap ST(t_r)$ or $u \in \mathsf{analz}((T_q \cup T) \setminus \{t_r\})$.

Now suppose $q > r$ and e_q is a receive event. We know that $t_q \in \overline{T_{q-1}}$. In fact $t_q \in \mathsf{synth}(\mathsf{analz}(T_{q-1}) \cap ST(t_q))$. Consider any $u \in \mathsf{analz}(T_{q-1}) \cap ST(t_q)$. For all analz-proofs π of $T_{q-1} \vdash u$ and for all $T_{q-1} \vdash t$ labelling the non-root nodes of π, (since ξ_{q-1} is not leaky) t is not secret at ξ_{q-2}. Hence we can apply the result of the previous paragraph to this case and conclude that $u \in (\mathsf{analz}(T_r) \cap ST(t_r)) \cup \mathsf{analz}((T_{q-1} \cup T) \setminus \{t_r\})$. If $u \in \mathsf{analz}((T_{q-1} \cup T) \setminus \{t_r\})$ we are done. Otherwise $u \in \mathsf{analz}(T_r) \cap ST(t_r)$. If now $EST(u) \neq \emptyset$ then since $u \in ST(t_r) \cap ST(t_q)$ it would follow that $EST(t_q) \cap EST(t_r) \neq \emptyset$ in contradiction to the fact that e_q is not a good successor of e_r. Thus $EST(u) = \emptyset$ and u is a tuple of atomic terms. In this case $u \in \mathsf{synth}(\mathsf{analz}(\{u\}) \cap \mathcal{T}_0)$. But then $\mathsf{analz}(\{u\}) \cap \mathcal{T}_0 \subseteq \mathsf{analz}(T_r) \cap \mathcal{T}_0 \subseteq \mathsf{analz}(T_{r-1} \cup T)$. This implies that $u \in \overline{(T_{q-1} \cup T) \setminus \{t_r\}}$. Thus we have proved that $\mathsf{analz}(T_{q-1}) \cap ST(t_q) \subseteq \overline{(T_{q-1} \cup T) \setminus \{t_r\}}$ and hence $t_q \in \overline{(T_{q-1} \cup T) \setminus \{t_r\}}$.

We are left with proving that ξ' is leaky. Since ξ is leaky (and $e_1 \cdots e_{k-1}$ is not), we can choose an analz-proof π whose root is labelled $T_k \vdash m$ for some m which is secret at $e_1 \cdots e_{k-1}$ and such that for all $T_k \vdash t$ labelling the non-root nodes of π, t is not secret at $e_1 \cdots e_{k-1}$. As observed earlier we can conclude that $m \in \mathsf{analz}((T_k \cup T) \setminus \{t_r\}) \cup \mathsf{analz}(T_r)$. Now we note that since m is secret at $e_1 \cdots e_{k-1}$, $m \notin \mathsf{analz}(T_r)$ (and thus $m \notin T$ as well). Therefore $m \in \mathsf{analz}((T_k \cup T) \setminus \{t_r\})$. Since $m \notin T$, it follows that $\tau(m) = m$. It can also be shown that $m \notin NT(e_r)$. Thus m is secret at $e_1 \cdots e_{r-1} e_{r+1} \cdots e_{k-1}$ as well. Therefore $\tau(m) = m$ is secret at $e'_1 \cdots e'_{r-1} e'_{r+1} \cdots e'_{k-1}$. Since $m \in \mathsf{analz}((T_k \cup T) \setminus \{t_r\})$ and since $\tau(T) = \{z\} \subseteq T_0$, it follows that $m = \tau(m) \in \mathsf{analz}(\tau(T_k \setminus \{t_r\}))$. Thus ξ' is leaky. This proves the reduction to good runs.

Bounding the length of good runs: We are left with proving that good runs are of bounded length, and further that it suffices to check a finite set of runs of Pr for leakiness. Suppose $\mathsf{Pr} = (\mathsf{C}, a_1 b_1 \cdots a_\ell b_\ell)$. Suppose ξ is some run of Pr and suppose that $e_1 \cdots e_r$ is a good path in ξ. The tagging scheme (specifically item 1 of Proposition 2.10) ensures that there exists a sequence $i_1 < \cdots < i_r \leq 2 \cdot \ell$ such that for all $j \leq r$, $act(e_j)$ is an instance of $act_{i_j}(\mathsf{Pr})$, where the notation $act_i(\mathsf{Pr})$ denotes $a_{(i+1)/2}$ if i is odd, and denotes $b_{i/2}$ if i is even. Thus all good paths in ξ are of length at most $2 \cdot \ell$. From item 2 of Proposition 2.10 (which is an immediate consequence of our tagging scheme), we see that there are at most two good predecessors in ξ. Putting these two facts together we can see that any good run is of length at most $2^{2 \cdot \ell + 1} - 1$. Now in any run of Pr of length bounded by B, only a bounded number of new nonces and keys are mentioned (the bound depending on the specification of Pr and B), apart from CT(Pr), the different K_A's and z, of course. They can all be uniformly renamed using terms in a fixed finite set T and thus it suffices to consider runs of Pr which are of bounded length and which refer to basic terms from T. Since the runs are well-typed the width of the terms occurring in the runs are also determined by the specification of Pr. Thus to check if a protocol has a *good well-typed leaky run* it suffices to check in a finite set of runs and thus this problem is decidable.

4 Discussion

We have followed a long chain of argument to show that detecting leaks in tagged protocols amounts to detecting leaks in a bounded set of runs, each of whose length is bounded. Here we have used a fixed tagging scheme. It is conceivable that many other tagging schemes would serve equally well. This raises the question of deciding whether a given well-formed protocol is 'taggable' or not (preserving leaks). If the only attacks on the protocol were type flaw attacks, tagging may be used to eliminate them and hence this question amounts to deciding whether the given protocol has non-type-flaw attacks, assuming that it has some attacks. This is an interesting issue not answered here. In ongoing work, we are also looking to extend the techniques used here to prove the decidability of security properties statable in a simple modal logic formalism.

References

1. Amadio, R.M., Lugiez, D. and Vanackère, V., "On the symbolic reduction of processes with cryptographic functions", *INRIA Research Report 4147*, March 2001.
2. Blanchet, B. and Podelski, P., "Verification of Cryptographic Protocols: Tagging Enforces Termination", In Goedon, A.D. ed. *Proc. FoSSaCS'03*, LNCS 2620, 2003, 136–152.
3. Comon-Lundh, H. and Cortier, V., "New decidability results for fragments of first-order logic and applications to cryptographic protocols", In Nieuwenhuis, R. ed. *Proc. RTA'2003*, LNCS 2706, 2003, 148–164.
4. Comon-Lundh, H., Cortier, V. and Mitchell, J.C., "Tree automata with One Memory, Set Constraints, and Ping-Pong Protocols", In *Proc. ICALP 2001*, LNCS 2076, 2001.
5. Clark, J. and Jacob, J., "A survey of authentication protocol literature", Electronic version available at `http://www.cs.york.ac.uk./~jac`, 1997.
6. Dolev, D., Even, S. and Karp, R.M., "On the Security of Ping-Pong Protocols", In *Information and Control*, 55, 1982, 57–68.
7. Durgin, N.A., Lincoln, P.D., Mitchell, J.C. and Scedrov, A., "The undecidability of bounded security protocols", In *Proc. FMSP'99*, 1999.
8. Heather, J., Lowe, G. and Schneider, S. "How to Prevent Type Flaw Attacks on Security Protocols", In *Proc. 13th IEEE CSFW*, 2000, 255-268.
9. Heintze, N. and Tygar, D., "A model for secure protocols and their composition", In *IEEE Transactions on Software Engineering*, 22, 1996, 16–30.
10. Lowe, G., "Towards a completeness result for model checking of security protocols", In *Journal of computer security*, 7, 1999, 89–146.
11. Millen, J.K. and Shmatikov, V., "Constraint solving for bounded-process cryptographic protocol analysis", In *Proc. ACM Conf. on Computer and Communications Security*, 2001, 166–175.
12. Paulson, L.C., "The inductive approach to verifying cryptographic protocols", In *Journal of computer security*, 6, 1998, 85–128.
13. Ramanujam, R. and Suresh, S.P., "A decidable subclass of unbounded security protocols", In Gorrieri, R. ed. *Proc. WITS'03*, April 2003, 11–20.
14. Ramanujam, R. and Suresh, S.P., "An equivalence on terms for security protocols", In Bharadwaj, R. ed. *Proc. AVIS'03*, April 2003, 45–56.
15. Ramanujam, R. and Suresh, S.P., "Decidability of secrecy for tagged protocols", `http://www.imsc.res.in/~jam`, September 2003.
16. Rusinowitch, M. and Turuani, M., "Protocol insecurity with finite number of sessions is NP-complete", In *Proc. CSFW 14*, 2001, 174–190.