Recursion vs. Replication in Simple Cryptographic Protocols

Hans Hüttel* and Jiří Srba**

BRICS* ***, Department of Computer Science, University of Aalborg
Fredrik Bajersvej 7B, 9220 Aalborg East, Denmark

Abstract. We use some recent techniques from process algebra to draw several conclusions about the well studied class of ping-pong protocols introduced by Dolev and Yao. In particular we show that all nontrivial properties, including reachability and equivalence checking wrt. the whole van Glabbeek's spectrum, become undecidable for a very simple recursive extension of the protocol. The result holds even if no nondeterministic choice operator is allowed. We also show that the extended calculus is capable of an implicit description of the active intruder, including full analysis and synthesis of messages in the sense of Amadio, Lugiez and Vanackère. We conclude by showing that reachability analysis for a replicative variant of the protocol becomes decidable.
Note: full proofs are available in [11].

1 Introduction

Process calculi have been suggested as a natural vehicle for reasoning about cryptographic protocols. In [1], Abadi and Gordon introduced the spi-calculus and described how properties such as secrecy and authenticity can be expressed via notions of observational equivalence (like may-testing). Alternatively, security questions have been studied using reachability analysis [3, 5, 9].

We provide a basic study of expressiveness and feasibility of cryptographic protocols. We are interested in two verification approaches: *reachability analysis* and *equivalence (preorder) checking*. In reachability analysis the question is whether a certain (bad or good) configuration of the protocol is reachable from a given initial one. In equivalence checking the question is whether a protocol implementation is equivalent (e.g. bisimilar) to a given specification (optimal behaviour). These verification strategies can be used even in the presence of an *active intruder* (in the Dolev-Yao style), i.e., an agent with capabilities to listen to any communication, to perform analysis and synthesis of communicated messages according to the actual knowledge of compromised keys, and to actively participate in the protocol behaviour by transmitting new messages. This can

* hans@cs.auc.dk
** srba@cs.auc.dk, supported in part by the GACR, grant No. 201/03/1161.
*** **B**asic **R**esearch in **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

be naturally implemented not only into the reachability analysis (see e.g. [4]) but also into the equivalence checking approach (see e.g. [10]).

A number of security properties are decidable for finite protocols [3, 14]. In the case of an unbounded number of protocol configurations, the picture is more complex. Durgin et al. showed in [8] that security properties are undecidable in a restricted class of so-called bounded protocols (that still allows for infinitely many reachable configurations). In [2] Amadio and Charatonik consider a language of tail-recursive protocols with bounded encryption depth and name generation; they show that, whenever certain restrictions on decryption are violated, one can encode two-counter machines in the process language. On the other hand, Amadio, Lugiez and Vanackère show in [4] that the reachability problem is in PTIME for a class of protocols with iteration.

In this paper we focus solely on ping-pong based behaviours of recursive and replicative protocols (perhaps the simplest behaviour of all studied calculi) in order to draw general conclusions about expressiveness and tractability of formal verification of cryptographic protocols. The class of *ping-pong protocols* was introduced in 1983 by Dolev and Yao [7]. The formalism deals with memory-less protocols which may be subjected to arbitrarily long attacks. Here, the secrecy of a finite ping-pong protocol can be decided in polynomial time. Later, Dolev, Even and Karp found a cubic-time algorithm [6]. The class of protocols studied in [4] contains iterative ping-pong protocols and, as a consequence, secrecy properties remain polynomially decidable even in this case.

In the present paper we continue our study of recursive and replicative extensions of ping-pong protocols. In [12] we showed that the recursive extension of the calculus is Turing powerful, however, the nondeterministic choice operator appeared to be essential in the construction. The question whether the calculus is Turing powerful even without any explicit way to define nondeterministic processes was left open. Here we present a radically new reduction from multi-stack automata and strengthen the undecidability results to hold even for protocols without nondeterministic choice. We prove, in particular, that both reachability and equivalence checking for all equivalences and preorders between trace equivalence/preorder and isomorphism of labelled transition systems (which includes all equivalences and preorders from van Glabbeek's spectrum [15]) become undecidable. These results are of general importance because they prove the impossibility of automated verification for essentially all recursive cryptographic protocols capable of at least the ping-pong behaviour.

In the initial study from [12], the question of active attacks on the protocol was not dealt with. We shall demonstrate that a complete notion of the active intruder (including analysis and synthesis of messages in the sense of Amadio, Lugiez and Vanackère [4]) can be explicitly encoded into our formalism.

Finally, we study a replicative variant of the calculus. Surprisingly, such a calculus becomes decidable, at least with regard to reachability analysis. We use a very recent result from process algebra (decidability of reachability for weak process rewrite systems by Křetínský, Řehák and Strejček [13]) in order to derive the result.

2 Basic definitions

2.1 Labelled transition systems with label abstraction

In order to provide a uniform framework for our study of ping-pong protocols, we define their semantics by means of labelled transition systems. A *labelled transition system* (LTS) is a triple $\mathcal{T} = (S, \mathcal{A}ct, \longrightarrow)$ where S is a set of *states* (or *processes*), $\mathcal{A}ct$ is a set of *labels* (or *actions*), and $\longrightarrow \subseteq S \times \mathcal{A}ct \times S$ is a *transition relation*, written $\alpha \xrightarrow{a} \beta$, for $(\alpha, a, \beta) \in \longrightarrow$. As usual we extend the transition relation to the elements of $\mathcal{A}ct^*$. We also write $\alpha \longrightarrow^* \beta$, whenever $\alpha \xrightarrow{w} \beta$ for some $w \in \mathcal{A}ct^*$.

The idea is that the states represent *global configurations* of a given protocol and the transitions describe the *information flow*. Labels on the transitions moreover represent the messages (both plain-text and cipher-text) which are being communicated during the state changes.

The explicit possibility to observe the full content of messages is sometimes not very realistic; it means that an external observer of such a system can e.g. distinguish between two different messages encrypted by the same encryption key, without the actual knowledge of the key.

In order to restrict capabilities of the observer we introduce a so called *label abstraction function* $\phi : \mathcal{A}ct \mapsto \mathcal{A}ct$. Given a LTS $\mathcal{T} = (S, \mathcal{A}ct, \longrightarrow_{\mathcal{T}})$ and a label abstraction function ϕ we define a new LTS $\mathcal{T}_\phi \stackrel{\text{def}}{=} (S, \mathcal{A}ct, \longrightarrow_{\mathcal{T}_\phi})$ where $\alpha \xrightarrow{\phi(a)}_{\mathcal{T}_\phi} \beta$ iff $\alpha \xrightarrow{a}_{\mathcal{T}} \beta$ for all $\alpha, \beta \in S$ and $a \in \mathcal{A}ct$. We call $\mathcal{T}_\phi$ a *labelled transition system with label abstraction*.

Let us now focus on the messages (actions). Assume a given set of encryption keys $\mathcal{K}$. The set of all messages over $\mathcal{K}$ is given by the following abstract syntax

$$m \ ::= \ k \ \mid \ k \cdot m$$

where k ranges over $\mathcal{K}$. Hence every element of the set $\mathcal{K}$ is a *(plain-text) message* and if m is a message then $k \cdot m$ is a *(cipher-text) message* (meaning that the message m is encrypted by the key k). Given a message $k_1 \cdot k_2 \cdots k_n$ over $\mathcal{K}$ we usually[1] write it only as a word $k_1 k_2 \cdots k_n$ from $\mathcal{K}^*$. Note that k_n is the plain-text part of the message and the outermost encryption key is always on the left (k_1 in our case). In what follows we shall identify the set of messages and $\mathcal{K}^*$, and we denote the extra element of $\mathcal{K}^*$ consisting of the empty sequence of keys by ϵ.

The level of abstraction we may select depends on the particular studied property we are interested in. Nevertheless, it seems reasonable to require at least the possibility to distinguish between plain-text and cipher-text messages. We say that a label abstraction function ϕ is *reasonable* iff $\phi(k) \neq \phi(k'w)$ for all $k, k' \in \mathcal{K}$ and $w \in \mathcal{K}^+$.

[1] In our previous work on ping-pong protocols [12] we denoted a message m encrypted by a key k as $\{m\}_k$. We changed the notation in order to improve the clarity of the proofs. In particular, when messages like $k_1 k_2 \cdots k_n$ are used, the previous syntax described the keys in a reversed order, which was technically inconvenient.

2.2 A calculus of recursive ping-pong protocols

We shall now define a calculus which captures exactly the class of ping-pong protocols by Dolev and Yao [7] extended (in a straightforward manner) with recursive definitions.

Let $\mathcal{K}$ be a set of encryption keys. A *specification* of a recursive ping-pong is a finite set of process definitions Δ such that for every *process constant* P (from a given set $\mathcal{C}onst$) the set Δ contains exactly one process definition of the form

$$P \stackrel{\text{def}}{=} \sum_{i_1 \in I_1} v_{i_1} \triangleright \, . \, \overline{w_{i_1} \triangleright} . P_{i_1} + \sum_{i_2 \in I_2} v_{i_2} . P_{i_2} + \sum_{i_3 \in I_3} \overline{w_{i_3}} . P_{i_3}$$

where I_1, I_2 and I_3 are finite sets of indices such that $I_1 \cup I_2 \cup I_3 \neq \emptyset$, and v_{i_1}, v_{i_2}, w_{i_1} and w_{i_3} are messages (belong to $\mathcal{K}^*$) for all $i_1 \in I_1$, $i_2 \in I_2$ and $i_3 \in I_3$, and $P_i \in \mathcal{C}onst \cup \{\mathbf{0}\}$ for all $i \in I_1 \cup I_2 \cup I_3$ such that $\mathbf{0}$ is a special constant called the *empty process*. We moreover require that v_{i_2} and w_{i_3} for all $i_2 \in I_2$ and $i_3 \in I_3$ are different from the empty message ϵ. (Observe that any specification Δ contains only finitely many keys.)

Summands continuing in the empty process constant $\mathbf{0}$ will be written without the $\mathbf{0}$ symbol and process definitions will often be written in their unfolded form using the *nondeterministic choice operator* '+'. An example of a process definition is e.g. $P \stackrel{\text{def}}{=} k_1 \triangleright \, . \, \overline{k_2 \triangleright} . P_1 + k_1 \triangleright \, . \, \overline{k_3 \triangleright} + k_1 k_2 . P_1 + \overline{k_1 k_1} + \overline{k_1 k_2} . P_2$.

The intuition is that each summand of the form $v_{i_1} \triangleright \, . \, \overline{w_{i_1} \triangleright} . P_{i_1}$ can receive a message encrypted by a sequence v_{i_1} of outermost keys, decrypt the message using these keys, send it out encrypted by the sequence of keys w_{i_1}, and finally behave as the process constant P_{i_1}. The symbol $\triangleright$ stands for the rest of the message after decrypting it with the key sequence v_{i_1}. This describes a standard ping-pong behaviour of the process. (Note that the symbol $\triangleright$ is equivalent to our $\{x\}$ notation from [12]).

In addition to this we may have summands of the forms $v_{i_2} . P_{i_2}$ and $\overline{w_{i_3}} . P_{i_3}$, meaning simply that a message is received and forgotten or unconditionally transmitted, respectively. This is a small addition to the calculus we presented in [12] in order to allow for discarding of old messages and generation of new messages. These two features were not available in the earlier version of the calculus but they appear to be technically convenient when modeling an explicit intruder and for strengthening the positive decidability results in Section 5. Nevertheless, the undecidability results presented in Section 3 are valid even without this extension since only the standard ping-pong behaviour is used in the constructions. A feature very similar to the forgetful input operation can be also found in [4].

A *configuration* of a ping-pong protocol specification Δ is a parallel composition of process constants, possibly preceded by output messages. Formally the set $\mathcal{C}onf$ of configurations is given by the following abstract syntax

$$C ::= \quad \mathbf{0} \quad | \quad P \quad | \quad \overline{w} . P \quad | \quad C \parallel C$$

where $\mathbf{0}$ is the empty configuration, $P \in \mathcal{C}onst \cup \{\mathbf{0}\}$ ranges over process constants including the empty process, $w \in \mathcal{K}^*$ ranges over the set of messages, and '$\|$' is the operator of parallel composition.

We introduce a structural congruence relation $\equiv$ which identifies configurations that represent the same state of the protocol. The relation $\equiv$ is defined as the least congruence over configurations ($\equiv \subseteq \mathcal{C}onf \times \mathcal{C}onf$) such that $(\mathcal{C}onf, \|, \mathbf{0})$ is a commutative monoid and $\overline{\epsilon}.P \equiv P$ for all $P \in \mathcal{C}onst$. In what follows we shall identify configurations up to structural congruence.

Remark 1. We let $\overline{\epsilon}.P \equiv P$ because the empty message should never be communicated. This means that when a prefix like $k\triangleright\,.\,\overline{\triangleright}.P$ receives a plain-text message k and tries to output $\overline{\epsilon}.P$, it simply continues as the process P.

We shall now define the semantics of ping-pong protocols in terms of labelled transition systems. We define a set $\mathcal{C}onf_S \subseteq \mathcal{C}onf$ consisting of all configurations that do not contain the operator of parallel composition and call these *simple configurations*. We also define two sets $In(C,m), Out(C,m) \subseteq \mathcal{C}onf_S$ for all $C \in \mathcal{C}onf_S$ and $m \in \mathcal{K}^+$. The intuition is that $In(C,m)$ ($Out(C,m)$) contains all configurations which can be reached from the simple configuration C after receiving (resp. outputting) the message m from (to) the environment. Formally, $In(C,m)$ and $Out(C,m)$ are the smallest sets which satisfy:

- $Q \in In(P,m)$ whenever $P \in \mathcal{C}onst$ and $m.Q$ is a summand of P
- $\overline{w\alpha}.Q \in In(P,m)$ whenever $P \in \mathcal{C}onst$ and $v\triangleright\,.\,\overline{w\triangleright}.Q$ is a summand of P such that $m = v\alpha$
- $P \in Out(\overline{m}.P, m)$ whenever $P \in \mathcal{C}onst \cup \{\mathbf{0}\}$
- $Q \in Out(P,m)$ whenever $P \in \mathcal{C}onst$ and $\overline{m}.Q$ is a summand of P.

A given protocol specification Δ determines a labelled transition system $T(\Delta) \stackrel{\text{def}}{=} (S, \mathcal{A}ct, \longrightarrow)$ where the states are configurations of the protocol modulo the structural congruence ($S \stackrel{\text{def}}{=} \mathcal{C}onf/\equiv$), the set of labels (actions) is the set of messages that can be communicated between the agents of the protocol ($\mathcal{A}ct \stackrel{\text{def}}{=} \mathcal{K}^+$), and the transition relation $\longrightarrow$ is given by the following SOS rule (recall that '$\|$' is commutative).

$$\frac{m \in \mathcal{K}^+ \qquad C_1, C_2 \in \mathcal{C}onf_S \qquad C_1' \in Out(C_1,m) \qquad C_2' \in In(C_2,m)}{C_1 \parallel C_2 \parallel C \;\stackrel{m}{\longrightarrow}\; C_1' \parallel C_2' \parallel C}$$

This means that (in the context C) two simple configurations (agents) C_1 and C_2 can communicate a message m in such a way that C_1 outputs m and becomes C_1' while C_2 receives the message m and becomes C_2'.

For further discussion and examples of recursive ping-pong protocols we refer the reader to [12].

2.3 Reachability and behavioural equivalences

One of the problems that is usually studied is that of *reachability analysis*: given two configurations $C_1, C_2 \in \mathcal{C}onf$ we ask whether C_2 is reachable from C_1, i.e., if $C_1 \longrightarrow^* C_2$. In this case the set of labels is irrelevant.

As the semantics of our calculus is given in terms of labelled transition systems (together with an appropriate label abstraction function), we can also study the *equivalence checking* problems. Given some behavioural equivalence or preorder $\leftrightarrow$ from van Glabbeek's spectrum [15] (e.g. strong bisimilarity or trace, failure and simulation equivalences/preorders just to mention a few) and two configurations $C_1, C_2 \in \mathcal{C}onf$ of a protocol specification Δ, the question is to decide whether C_1 and C_2 are $\leftrightarrow$-equivalent (or $\leftrightarrow$-preorder related) in $T(\Delta)$, i.e., whether $C_1 \leftrightarrow C_2$.

3 Recursive ping-pong protocols without explicit choice

In this section we strengthen the undecidability result from [12] and show that the reachability and equivalence checking problems are undecidable for ping-pong protocols without an explicit operator of nondeterminism and using classical ping-pong behaviour only, i.e., for protocols without any occurrence of the choice operator '+' and where every defining equation is of the form $P \stackrel{\text{def}}{=} v\triangleright\,.\,\overline{w\triangleright}.P'$ such that $P' \in \mathcal{C}onst$.

We moreover show that the negative results apply to all behavioural equivalences and preorders between trace equivalence/preorder and isomorphism of LTS (which preserves labelling) with regard to all reasonable label abstraction functions as defined in Section 2.

These results are achieved by showing that recursive ping-pong protocols can step-by-step simulate a Turing powerful computational device, in our case a computational model called multi-stack machines.

A *multi-stack machine* R with ℓ stacks ($\ell \geq 1$) is a triple $R = (Q, \Gamma, \longrightarrow)$ where Q is a finite set of *control-states*, Γ is a finite *stack alphabet* such that $Q \cap \Gamma = \emptyset$, and $\longrightarrow \subseteq Q \times \Gamma \times Q \times \Gamma^*$ is a finite set of *transition rules*, written $pX \longrightarrow q\alpha$ for $(p, X, q, \alpha) \in \longrightarrow$.

A *configuration* of a multi-stack machine R is an element from $Q \times (\Gamma^*)^\ell$. We assume a given initial configuration $(q_0, w_1, \ldots, w_\ell)$ where $q_0 \in Q$ and $w_i \in \Gamma^*$ for all i, $1 \leq i \leq \ell$. If some of the stacks w_i are empty, we denote them by ϵ.

A *computational step* is defined such that whenever there is a transition rule $pX \longrightarrow q\alpha$ then a configuration which is in the control-state p and has X on top of the i'th stack (the tops of the stacks are on the left) can perform the following transition: $(p, w_1, \ldots, Xw_i, \ldots, w_\ell) \longrightarrow (q, w_1, \ldots, \alpha w_i, \ldots, w_\ell)$ for all $w_1, \ldots, w_\ell \in \Gamma^*$ and for all i, $1 \leq i \leq \ell$.

It is a folklore result that multi-stack machines are Turing powerful. Hence (in particular) the following problem is easily seen to be undecidable: given an initial configuration $(q_0, w_1, \ldots, w_\ell)$ of a multi-stack machine R, can we reach the configuration $(h, \epsilon, \ldots, \epsilon)$ for a distinguished halting control-state $h \in Q$ such

that all stacks are empty? Without loss of generality we can even assume that a configuration in the control-state h is reachable iff all stacks are empty.

Let $R = (Q, \Gamma, \longrightarrow)$ be a multi-stack machine. We define the following set of keys of a ping-pong specification Δ: $\mathcal{K} \stackrel{\text{def}}{=} Q \cup \Gamma \cup \{k_p \mid p \in Q\} \cup \{t, k_*\}$. Here t is a special key such that every communicated message is an encryption of the plain-text key t. The reason for this is that it ensures that the protocol never communicates any plain-text message. The key k_* is a special purpose locking key and it is explained later on in the construction.

We shall construct a ping-pong protocol specification Δ as follows.

- For every transition rule $pX \longrightarrow q\alpha$ we have a process constant $P_{pX \longrightarrow q\alpha}$ with the following defining equation: $P_{pX \longrightarrow q\alpha} \stackrel{\text{def}}{=} pX\rhd \,.\, \overline{k_q \alpha \rhd}.P_{pX \longrightarrow q\alpha}$.
- For every state $p \in Q$ we have two process constants T_p and T'_p.

$$T_p \stackrel{\text{def}}{=} k_p\rhd \,.\, \overline{k_*\rhd}.T'_p$$

$$T'_p \stackrel{\text{def}}{=} k_*\rhd \,.\, \overline{p\rhd}.T_p \quad \text{if } p \in Q \setminus \{h\}, \text{ and} \quad T'_h \stackrel{\text{def}}{=} h\rhd \,.\, \overline{h\rhd}.T'_h$$

  Recall that $h \in Q$ is the halting control-state.
- Finally, we define a process constant B (standing for a buffer over a fixed key k_*): $B \stackrel{\text{def}}{=} k_*\rhd \,.\, \overline{k_*\rhd}.B$.
  In this defining equation the key k_* locks the content of the buffer such that it is accessible only by some T'_p.

Note that Δ does not contain any choice operator '+' as required.

Let $(q_0, w_1, \ldots, w_\ell)$ be an initial configuration of the multi-stack machine R. The corresponding initial configuration of the protocol Δ is defined as follows (the meta-symbol Π stands for a parallel composition of the appropriate components).

$$\Big(\prod_{(r,A,s,\beta)\in\longrightarrow} P_{rA\longrightarrow s\beta}\Big) \;\|\; \Big(\prod_{p\in Q\setminus\{q_0\}} T_p\Big) \;\|\; T'_{q_0} \;\|\; \Big(\prod_{j\in\{1,\ldots,\ell\}} \overline{k_* w_j t}.B\Big) \quad (1)$$

The following invariants are preserved during any computational sequence starting from this initial configuration:

- at most one T'_p for some $p \in Q$ is present as a parallel component (the intuition is that this represents the fact that the machine R is in the control-state p), and
- plain-text messages are never communicated.

Theorem 1. *The reachability problem for recursive ping-pong protocols without an explicit choice operator is undecidable.*

Theorem 2. *The equivalence checking problem for recursive ping-pong protocols without an explicit choice operator is undecidable for any behavioral equivalence/preorder between trace equivalence/preorder and isomorphism (including all equivalences and preorders from van Glabbeek's spectrum [15]) and for any reasonable label abstraction function.*

4 The active intruder

In the literature on applying process calculi to the study of cryptographic protocols, there have been several proposals for explicit modelling the active intruder (environment). Foccardi, Gorrieri and Martinelli in [10] express the environment within the process calculus, namely as a process running in parallel with the protocol. In [4] Amadio, Lugiez and Vanackère describe a tiny process calculus similar to ours, except that they use replication instead of recursion. Moreover, the environment is described in the semantics of the calculus. Transitions are of the form $(C, T) \to (C', T')$ where C and C' are protocol configurations and T and T' denote the sets of messages known to the environment (all communication occurs only by passing messages through these sets).

The environment is assumed to be hostile; it may compute new messages by means of the operations of analysis and synthesis and pass these on to the process. Let $\mathcal{K}$ be a set of encryption keys as before. The *analysis* of a set of messages $T \subseteq \mathcal{K}^*$ is the least set $\mathcal{A}(T)$ satisfying

$$\mathcal{A}(T) = T \cup \{w \mid kw \in \mathcal{A}(T),\ k \in \mathcal{K} \cap \mathcal{A}(T)\}. \tag{2}$$

The *synthesis* of a set of messages $T \subseteq \mathcal{K}^*$ is the least set $\mathcal{S}(T)$ satisfying

$$\mathcal{S}(T) = \mathcal{A}(T) \cup \{kw \mid w \in \mathcal{S}(T),\ k \in \mathcal{K} \cap \mathcal{S}(T)\}. \tag{3}$$

We can now design an environment sensitive semantics for our calculus close in style to that of [4]. We define the reduction relation $\to$ by the following set of axioms (here $x \in P$ means that x is a summand in the defining equation of the process constant P).

$$\begin{array}{lll}
(P \parallel C, T) \to (\overline{w\alpha}.P' \parallel C, T) & \text{if } (v\triangleright\,.\,\overline{w\triangleright}.P') \in P \text{ and } v\alpha \in \mathcal{S}(T) & \text{(A1)} \\
(P \parallel C, T) \to (P' \parallel C, T) & \text{if } (v.P') \in P \text{ and } v \in \mathcal{S}(T) & \text{(A2)} \\
(\overline{w}.P \parallel C, T) \to (P \parallel C, T \cup \{w\}) & & \text{(A3)} \\
(P \parallel C, T) \to (P' \parallel C, T \cup \{w\}) & \text{if } (\overline{w}.P') \in P & \text{(A4)}
\end{array}$$

We show that this semantics can be internalized in our calculus within our existing semantics. The construction is nontrivial as (on the contrary with stronger calculi like spi-calculus) we can use only a very limited set of operations. The details are in the full paper.

Theorem 3. *For any recursive ping-pong protocol, we can define its new parallel component which enables all the attacks described by axioms (A1) – (A4).*

5 Replicative ping-pong protocols

In this section we shall define a replicative variant of our calculus for ping-pong protocols. We will then show that this formalism is not Turing powerful because the reachability problem becomes decidable.

Let us now define *replicative ping-pong protocols*. Let $\mathcal{K}$ be the set of encryption keys as before. The set $\mathcal{C}onf$ of protocol configurations is given by the following abstract syntax

$$C \ ::= \ \mathbf{0} \ \mid \ v\triangleright \,.\, \overline{w\triangleright} \ \mid \ v \ \mid \ \overline{w} \ \mid !(v\triangleright \,.\, \overline{w\triangleright}) \ \mid !(v) \ \mid !(\overline{w}) \ \mid \ C \parallel C$$

where $\mathbf{0}$ is the symbol for the empty configuration, v and w range over $\mathcal{K}^*$, and ! is the bang operator (replication). As before, we shall introduce structural congruence $\equiv$, which is the smallest congruence over $\mathcal{C}onf$ such that $(\mathcal{C}onf, \parallel, \mathbf{0})$ is a commutative monoid; $\epsilon \parallel C \equiv C \equiv \overline{\epsilon} \parallel C$; $!(\epsilon) \equiv \mathbf{0} \equiv !(\overline{\epsilon})$; and $!(C) \equiv C \parallel !(C)$. A labelled transition system determined by a configuration (where states are configurations modulo $\equiv$ and labels are non-empty messages as before) is defined by the following SOS rules (recall the replicative axiom $!(C) \equiv C \parallel !(C)$ and the fact that '$\parallel$' is commutative).

$$\frac{m \in \mathcal{K}^+}{\overline{m} \parallel m \parallel C \ \stackrel{m}{\longrightarrow} \ C} \qquad \frac{m \in \mathcal{K}^+ \qquad m = v\alpha}{\overline{m} \parallel v\triangleright \,.\, \overline{w\triangleright} \parallel C \ \stackrel{m}{\longrightarrow} \ \overline{w\alpha} \parallel C}$$

We can now show that the reachability problem for general replicative ping-pong protocols is decidable. We reduce our problem to reachability of weak process rewrite systems (wPRS) which was very recently proven to be decidable [13].

Theorem 4. *The reachability problem for replicative ping-pong protocols is decidable.*

6 Conclusion

We have seen that ping-pong protocols extended with recursive definitions have full Turing power. This is the case even in the absence of nondeterministic choice operator '+'. A result like this implies that any reasonable property for all richer calculi cannot be automatically verified.

We also presented an explicit description of the active intruder in the syntax of recursive ping-pong protocols.

Finally, we showed that reachability analysis for a replicative variant of the protocol becomes feasible. Our proof uses very recent results from process algebra [13] and can be compared to the work of Amadio, Lugiez and Vanackère [4] which establishes the decidability of reachability for a similar replicative protocol capable of ping-pong behaviour. Their approach uses a notion of a pool of messages explicitly modelled in the semantics and reduces the question to a decidable problem of reachability for prefix rewriting. In our approach we allow spontaneous generation of new messages which is not possible in their calculus. Moreover, we can distinguish between replicated and once-only behaviours (unlike in [4] where all processes have to be replicated).

Last but not least we hope that our approach can be possibly extended to include other operations as the decidability result for replicative protocols uses

only a limited power of wPRS (only a parallel composition of stacks). Hence there is a place for further extensions of the protocol syntax while preserving a decidable calculus (e.g. messages of the form $\overline{k_1(k_2\ op\ k_3)k_4}$ for some extra composition operation *op* on keys can be easily stored in wPRS as $k_1.(k_2 \parallel k_3).k_4$). Such a study is left for future research.

References

1. M. Abadi and A.D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.
2. R.M. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proc. of CONCUR'02*, vol. 2421 of *LNCS*, 499–514. Springer-Verlag, 2002.
3. R.M. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proc. of CONCUR'00*, vol. 1877 of *LNCS*, 380–394. Springer-Verlag, 2000.
4. R.M. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *TCS*, 290(1):695–740, October 2002.
5. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. of ICALP'01*, vol. 2076 of *LNCS*, 667–681. Springer, 2001.
6. D. Dolev, S. Even, and R.M. Karp. On the security of ping-pong protocols. *Information and Control*, 55(1–3):57–68, 1982.
7. D. Dolev and A.C. Yao. On the security of public key protocols. *Transactions on Information Theory*, IT-29(2):198–208, 1983.
8. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In N. Heintze and E. Clarke, editors, *Proc. of FMSP'99*, 1999.
9. M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proc. of CSFW'01*, 160–173. IEEE, 2001.
10. R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *Proc. of ICALP'00*, vol. 1853 of *LNCS*, 354–372. Springer-Verlag, 2000.
11. H. Hüttel and J. Srba. Recursion vs. replication in simple cryptographic protocols. Technical Report RS-04-23, BRICS Research Series, 2004.
12. H. Hüttel and J. Srba. Recursive ping-pong protocols. In *Proc. of WITS'04*, 129–140, 2004.
13. M. Křetínský, V. Řehák, and J. Strejček. Extended process rewrite systems: Expressiveness and reachability. In *Proc. of CONCUR'04*, vol. 3170 of *LNCS*, 355–370. Springer-Verlag, 2004.
14. M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete. *TCS*, 299, 2003.
15. R.J. van Glabbeek. The linear time - branching time spectrum I: The semantics of concrete, sequential processes. In *Handbook of Process Algebra*, chapter 1, 3–99. Elsevier Science, 2001.