Decidability Issues for Extended Ping-Pong Protocols

Hans Hüttel (hans@cs.aau.dk) and Jiří Srba (srba@cs.aau.dk) *
BRICS†, *Department of Computer Science, University of Aalborg*
Fredrik Bajersvej 7B, 9220 Aalborg East, Denmark

May 19, 2005

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 but reachability is shown to be decidable in polynomial time if only two parties are participating in the protocol. We also show that the 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.

Keywords: cryptographic protocols, formal modelling, recursion, replication

1. Introduction

Process calculi have been suggested as a natural vehicle for reasoning about cryptographic protocols. In (Abadi and Gordon, 1998), Abadi and Gordon introduced the spi-calculus (a variant of the π-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 (Amadio and Lugiez, 2000; Boreale, 2001; Fiore and Abadi, 2001).

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*

* The author is supported in part by grants ITI-1M0021620808, 1ET-408050503, and GACR 201/03/1161.

† **B**asic **R**esearch in **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

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 be naturally implemented not only into the reachability analysis (see e.g. (Amadio et al., 2002)) but also into the equivalence checking approach. Within the equivalence (preorder) checking approach, correctness may be expressed as follows (following (Focardi et al., 2000)): "a protocol P guarantees a security property X if, whatever hostile environment E with a certain initial knowledge ϕ_I, then P is equivalent (in preorder) to (with) the specification $\alpha(P)$." Formally this is given by saying that

$$\text{protocol P satisfies property } X \quad \text{iff} \quad \forall E \in \mathcal{E} : \; P \parallel E \approx \alpha(P). \quad (1)$$

By an appropriate choice of the specification function α and a suitable equivalence (preorder) $\approx$, several security properties can be verified. Here is a small selection:

- *Secrecy* (confidential information should be available only to the partners of the communication). Here $\approx$ stands for trace preorder.
- *(Message) authenticity* (identification of other agents (messages) participating in communication). Here $\approx$ stands for trace equivalence or preorder.
- *Fairness* (in a contract, no party can gain advantage by ending the protocol prematurely). Here $\approx$ stands for failure equivalence.

Various notions of bisimilarity are studied in this context as bisimilarity is usually the "most decidable behavioral equivalence" which was confirmed e.g. by several positive decidability results in process algebra (Burkart et al., 2001). Hence the questions whether a certain class of cryptographic protocols has decidable reachability and equivalence (bisimilarity) checking are of particular importance for automated verification.

A number of security properties are decidable for finite protocols (Amadio and Lugiez, 2000; Rusinowitch and Turuani, 2003). In the case of an unbounded number of protocol configurations, the picture is more complex. Durgin et al. showed in (Durgin et al., 1999) that security properties are undecidable in a restricted class of so-called bounded protocols (that still allows for infinitely many reachable configurations). In (Amadio and Charatonik, 2002) 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 (Amadio et al., 2002) 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 (Dolev and Yao, 1983). 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 (Dolev et al., 1982). The class of protocols studied in (Amadio et al., 2002) contains iterative ping-pong protocols and, as a consequence, secrecy properties remain polynomially decidable even in this case.

In the present paper we study recursive and replicative extensions of ping-pong protocols. In (Hüttel and Srba, 2004) 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 different 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 (van Glabbeek, 2001)) 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. We also show that under the assumption that only two parties participate in the protocol, the reachability problem becomes decidable in polynomial time.

In the initial study from (Hüttel and Srba, 2004), 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 (Amadio et al., 2002)) can be explicitly encoded into our formalism in order to analyze general properties like in the scheme (1).

Finally, we study a replicative variant of the calculus. Surprisingly, such a calculus becomes decidable, at least with regard to the reachabil-

ity analysis. We use a very recent result from process algebra (decidability of reachability for weak process rewrite systems by Křetínský, Řehák and Strejček (Křetínský et al., 2004)) in order to derive the result. Our positive result for the replicative calculus is a formal confirmation of the general trend of using replication instead of recursion in process calculi for cryptographic protocols and explains why recursion is hardly ever considered in this setting.

*This paper is a revised and unified version of (*Hüttel and Srba, 2004*) and (*Hüttel and Srba, 2005*).*

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.

Remark 1. In (Hüttel and Srba, 2004) the semantics of ping-pong protocols is given in terms of transition systems with knowledge, i.e., unlabelled transition systems where each state it assigned its knowledge, represented as a subset of a certain set of all possible knowledge values. By standard techniques such a knowledge-based semantics can be translated to labelled transition systems and the studied verification properties (reachability, equivalence checking, etc.) are preserved. For example a state A with two knowledge values p_1 and p_2 can be transformed to a labelled transition system where the values p_1 and p_1 are represented as self-loops in state A which are visible under special actions p_1 and p_2. A fresh action a is used to represent the change of the state (the unlabelled transitions in the original knowledge-based semantics).

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 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.

Example 1. Let us consider a labelled transition system

$$\mathcal{T} \stackrel{\text{def}}{=} (S, \mathcal{A}ct, \longrightarrow)$$

where $S \stackrel{\text{def}}{=} \{A, B, C\}$, $\mathcal{A}ct \stackrel{\text{def}}{=} \mathcal{K}^*$ for a given set of keys $\mathcal{K} = \{k_1, k_2, \lambda\}$ and $\longrightarrow$ is given by the following picture.

$$A \xrightarrow{k_1 k_2} B \xrightarrow{k_2} C$$

The protocol computation starts in the state A and is very simple. First a plain-text k_2 encrypted by the encryption key k_1 is communicated to the process B, which decrypts the message and sends out the plain-text k_2. Let us now assume a label abstraction function ϕ defined by $\phi(k) = k$ if $k \in \mathcal{K}$ and $\phi(m) = \lambda$ otherwise. The labelled transition system T_ϕ with label abstraction function ϕ now looks as follows.

$$A \xrightarrow{\lambda} B \xrightarrow{k_2} C$$

This translates to the fact that the external observer is not allowed to see the content of encrypted messages (the action λ is used instead) and only plain-text messages can be recognized. □

The level of abstraction we may select depends on the particular studied property we are interested in and it directly corresponds to the specification function α from (1). 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 $w \in \mathcal{K}^+$.

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 (Dolev and Yao, 1983) 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

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

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.

$$P \stackrel{\text{def}}{=} k_1 \rhd . \overline{k_2 \rhd}.P_1 + k_1 \rhd . \overline{k_3 \rhd} + k_1 k_2.P_1 + \overline{k_1 k_1} + \overline{k_1 k_2}.P_2.$$

The intuition is that each summand of the form $v_{i_1} \rhd . \overline{w_{i_1} \rhd}.P_{i_1}$ 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 $\rhd$ 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.

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 (Hüttel and Srba, 2004) 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 3.1 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 (Amadio et al., 2002).

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 '$\parallel$' 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 $(\mathcal{C}onf, \parallel, \mathbf{0})$ 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 2. 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 $m = v\alpha$

- $P \in \mathit{Out}(\overline{m}.P, m)$ whenever $P \in \mathcal{C}\mathit{onst} \cup \{\mathbf{0}\}$
- $Q \in \mathit{Out}(P, m)$ whenever $P \in \mathcal{C}\mathit{onst}$ 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 ($S \stackrel{\text{def}}{=} \mathcal{C}\mathit{onf}/\equiv$), 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).

$$\frac{m \in \mathcal{K}^+ \qquad C_1, C_2 \in \mathcal{C}\mathit{onf}_S \qquad C_1' \in \mathit{Out}(C_1, m) \qquad C_2' \in \mathit{In}(C_2, m)}{C_1 \parallel C_2 \parallel C \xrightarrow{m} C_1' \parallel C_2' \parallel C}$$

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'.

Example 2. Let us consider a protocol specification Δ.

$$P \stackrel{\text{def}}{=} \overline{k}.P \;+\; k\triangleright\,.\,\overline{kk\triangleright}.P \;+\; k.Q \qquad\qquad Q \stackrel{\text{def}}{=} k.P$$

A fragment of the labelled transition system reachable from the initial configuration $P \parallel P$ looks as follows.

$$\begin{array}{ccccccc} P \parallel P & \xrightarrow{\quad k \quad} & P \parallel \overline{kk}.P & \xrightarrow{\quad kk \quad} & P \parallel \overline{kkk}.P & \xrightarrow{\quad kkk \quad} & \cdots \\ {\scriptstyle k}\big\downarrow \quad \big\downarrow{\scriptstyle k} & & & & & & \\ P \parallel Q & & & & & & \end{array}$$

□

2.3. Reachability and behavioural equivalences

One of the problems that is usually studied is that of *reachability analysis*: given two configurations $C_1, C_2 \in \mathcal{C}\mathit{onf}$ 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 (van Glabbeek, 2001) (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. Decidability Issues

In this section we shall discuss decidability questions for recursive ping-pong protocols. First we demonstrate a negative result for protocols with arbitrary many participants and then we show that reachability is decidable in polynomial time if we restrict ourself to protocols where only two parties are involved.

3.1. Recursive ping-pong protocols without explicit choice

In what follows we strengthen the undecidability result from (Hüttel and Srba, 2004) 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 $P' \in \mathcal{C}onst$.

Remark 3. Note that every process constant is allowed to have exactly one defining equation, however, no constraints are imposed on the communication behaviour of the parallel components.

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.

$$P_{pX \longrightarrow q\alpha} \stackrel{\text{def}}{=} pX\triangleright \,.\, \overline{k_q \alpha \triangleright}.P_{pX \longrightarrow q\alpha}$$

- For every state $p \in Q$ we have two process constants T_p and T'_p.

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

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

  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_*).

$$B \stackrel{\text{def}}{=} k_*\triangleright \,.\, \overline{k_*\triangleright}.B$$

  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).

$$\Big(\prod_{(r,A,s,\beta)\in\longrightarrow} P_{rA\longrightarrow s\beta}\Big) \parallel \Big(\prod_{p\in Q\setminus\{q_0\}} T_p\Big) \parallel T'_{q_0} \parallel \Big(\prod_{j\in\{1,\ldots,\ell\}} \overline{k_* w_j t}.B\Big) \quad (2)$$

The following invariants will be 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.

Let $(p, w_1, \ldots, w_i, \ldots w_\ell) \longrightarrow (q, w_1, \ldots, \alpha w'_i, \ldots w_\ell)$ be a computational step of R using the rule $pX \longrightarrow q\alpha$ such that $w_i = Xw'_i$. This one step is simulated by a sequence of four transitions in the ping-pong protocol Δ (see Figure 3.1). In the first step one buffer is selected and unlocked (the current control-state p replaces the locking key k_* in the outermost encryption). In particular the buffer $\overline{k_* w_i t}.B$ can be unlocked. No other kinds of transitions are possible in the first step. Opening of the selected buffer means that some of the process constants $P_{rA\longrightarrow s\beta}$ become able to accept this message. In particular the process constant $P_{pX\longrightarrow q\alpha}$ can receive the message and output $\overline{k_q \alpha w_i t}$ for further communication; the key k_q determines the control-state change. (At this stage also a communication between $\overline{k_* w_i t}.B$ and B is enabled but it does not change the current state and hence it cannot contribute to a computational progress.) In the next step only T_q can receive the message $\overline{k_q \alpha w_i t}$, it remembers the new control-state q by becoming T'_q and offers the k_*-locked message for a communication with B. This last communication (when B receives back the modified buffer) ends a simulation of one computational step of R.

The following property is easy to see: if after some number of steps starting in a protocol configuration corresponding to $(p, w_1, \ldots, w_\ell)$ we reach a first protocol configuration where T'_q appears for some $q \in Q$ then this corresponds to one correct computational step in R. On the other hand, the computation of Δ can get stuck after the first communication step (in case that the unlocked buffer does not enable an application of any rule $rA \longrightarrow s\beta$) or an infinite sequence

$$\Big(\prod_{(r,A,s,\beta)\in\longrightarrow} P_{rA\longrightarrow s\beta}\Big) \parallel \Big(\prod_{r\in Q\setminus\{p\}} T_r\Big) \parallel T'_p \parallel \Big(\prod_{j\in\{1,\ldots,\ell\}} \overline{k_* w_j t}.B\Big)$$

$$\downarrow k_* w_i t$$

$$\Big(\prod_{(r,A,s,\beta)\in\longrightarrow} P_{rA\longrightarrow s\beta}\Big) \parallel \Big(\prod_{r\in Q\setminus\{p\}} T_r\Big) \parallel \overline{p w_i t}.T_p \parallel \Big(\prod_{j\in\{1,\ldots,\ell\},j\neq i} \overline{k_* w_j t}.B\Big) \parallel B$$

$$\downarrow p w_i t$$

$$\Big(\prod_{(r,A,s,\beta)\in(\longrightarrow\setminus\{(p,X,q,\alpha)\})} P_{rA\longrightarrow s\beta}\Big) \parallel \overline{k_q \alpha w'_i t}.P_{pX\longrightarrow q\alpha} \parallel \Big(\prod_{r\in Q} T_r\Big) \parallel \Big(\prod_{j\in\{1,\ldots,\ell\},j\neq i} \overline{k_* w_j t}.B\Big) \parallel B$$

$$\downarrow k_q \alpha w'_i t$$

$$\Big(\prod_{(r,A,s,\beta)\in\longrightarrow} P_{rA\longrightarrow s\beta}\Big) \parallel \Big(\prod_{r\in Q\setminus\{q\}} T_r\Big) \parallel \overline{k_* \alpha w'_i t}.T'_q \parallel \Big(\prod_{j\in\{1,\ldots,\ell\},j\neq i} \overline{k_* w_j t}.B\Big) \parallel B$$

$$\downarrow k_* \alpha w'_i t$$

$$\Big(\prod_{(r,A,s,\beta)\in\longrightarrow} P_{rA\longrightarrow s\beta}\Big) \parallel \Big(\prod_{r\in Q\setminus\{q\}} T_r\Big) \parallel T'_q \parallel \Big(\prod_{j\in\{1,\ldots,\ell\},j\neq i} \overline{k_* w_j t}.B\Big) \parallel \overline{k_* \alpha w'_i t}.B$$

Figure 1. Simulation of $(p, w_1, \ldots, w_i, \ldots w_\ell) \longrightarrow (q, w_1, \ldots, \alpha w'_i, \ldots w_\ell)$ s.t. $w_i = X w'_i$

of communication steps of the form $\overline{m}.B \parallel B \xrightarrow{m} \overline{m}.B \parallel B$ is also possible.

This is formally captured in the following lemma.

Lemma 1. In the multi-stack machine R it holds that the configuration $(q, w'_1, \ldots, w'_\ell)$ is reachable from $(p, w_1, \ldots, w_\ell)$ if and only

if

$$\Big(\prod_{(r,A,s,\beta)\in\longrightarrow} P_{rA\longrightarrow s\beta}\Big) \parallel \Big(\prod_{p\in Q\setminus\{q\}} T_p\Big) \parallel T'_q \parallel \Big(\prod_{j\in\{1,\ldots,\ell\}} \overline{k_* w'_j t}.B\Big)$$

is reachable (in Δ) from

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

The following theorems are now easily derived.

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

Proof. Immediately from Lemma 1 and from the undecidability of reachability for multi-stack machines. □

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 (van Glabbeek, 2001)) and for any reasonable label abstraction function.

Proof. Let R be a multi-stack machine and Δ the protocol specification constructed above with the initial configuration C as given by (2). We consider the question whether $C \parallel \overline{h}$ is equivalent (or in preorder) with C.

In case that the halting control-state h is not reachable from the initial configuration of R, we know from Lemma 1 that T'_h will never appear as a parallel component in any reachable state from C. This implies that the plain-text message $\overline{h}$ will never be communicated and hence $C \parallel \overline{h}$ and C exhibit isomorphic behaviours under any label abstraction function.

On the other hand, if h is reachable from the initial configuration of R then because of Lemma 1 a configuration in Δ with the parallel component T'_h is reachable. Such a configuration is stuck in the process on the right, however, in the process on the left the plain-text message h can be communicated between T'_h and the extra parallel component $\overline{h}$. This means that $C \parallel \overline{h}$ and C are not even related by the trace preorder (and hence they are also not trace equivalent) because after a finite sequence of communicated messages there is a successor of the configuration $C \parallel \overline{h}$ which can communicate the plain-text h while

(as argued before) C can only exchange cipher-text messages. As the label abstraction function ϕ is reasonable, necessarily for all messages m (cipher-texts) communicated in C it is the case that $\phi(m) \neq \phi(h)$.

To sum up, if the machine R cannot reach the halting configuration then $C \parallel \overline{h}$ and C are isomorphic and if R halts than $C \parallel \overline{h}$ and C are not in trace preorder. This implies that all equivalences and preorders between trace and isomorphism are undecidable for any reasonable label abstraction function. □

The arguments in the proof of Theorem 2 imply that secrecy for recursive ping-pong protocols (i.e. the question whether a plain text message is ever directly communicated among the protocol participants) is also undecidable. The claim is valid even without the presence of an active intruder.

As mentioned in the introduction, protocol properties such as secrecy, authenticity and fairness can be expressed also within the equivalence checking approach (Focardi et al., 2000; Abadi and Gordon, 1998), considering preorders and equivalences like trace preorder, trace equivalence and failure equivalence. The result of Theorem 2 implies that for recursive ping-pong protocols, none of the protocol properties in the equivalence checking setting are decidable.

3.2. Protocols with Two Participants

If the family of protocols we consider contains at most two participants (parallel components) in every reachable configuration, we get a class similar to that of the traditional ping-pong protocols (Dolev et al., 1982; Dolev and Yao, 1983). In fact, our class is more general in the sense that we allow for recursive definitions in the protocol specification. In the situation of at most two parallel components in any reachable configuration we may without loss of generality assume that such configurations are always of the form $P \parallel \overline{w}.Q$ for some $P \in \mathcal{C}onst$, $Q \in \mathcal{C}onst \cup \{\mathbf{0}\}$ and $w \in \mathcal{K}^*$. If this is not the case, the computation of such a protocol is stuck — no communication can take place. We also assume that all process definitions are of the from $P \stackrel{\text{def}}{=} \sum_{i \in I} v_i \triangleright . \overline{w_i \triangleright} . P_i$, i.e., there are no summands of the form $v.Q$ or $\overline{w}.Q$ (these summands will also make the computation of the protocol get stuck as we obtain a configuration with two parallel components that both either try to input a message or output a message).

We shall now demonstrate that the class of ping-pong protocols with two participants is not Turing powerful since reachability now becomes decidable. Hence we can still hope for automatic verification of some protocol properties.

Theorem 3. The reachability problem for ping-pong protocols with two participants is decidable in polynomial time.

Proof. We reduce reachability of ping-pong protocols with two participants to reachability of pushdown automata (PDA), disregarding the input alphabet (see e.g. (Büchi, 1964)). In fact, we will use a slightly more general notion of PDA where several stack symbols can be removed in one computational step.

Let Δ be a given protocol specification and let $P_1 \parallel \overline{w_1}.Q_1$ and $P_2 \parallel \overline{w_2}.Q_2$ be two configurations of the protocol. We shall construct a PDA system together with two configurations $p_1\alpha_1$ and $p_2\alpha_2$ such that $P_1 \parallel \overline{w_1}.Q_1 \longrightarrow^* P_2 \parallel \overline{w_2}.Q_2$ if and only if $p_1\alpha_1 \longrightarrow^* p_2\alpha_2$.

The set of control states of the PDA automaton is $\{(P,Q) \mid P,Q \in \mathcal{C}onst \cup \{\mathbf{0}\}\}$ and the stack alphabet is equal to the set of the encryption keys $\mathcal{K}$. We have now a natural correspondence between configurations of the protocol and those of the PDA system. A protocol configuration $P \parallel \overline{w}.Q$ corresponds to a PDA configuration $(P,Q)w$ such that (P,Q) is the control state (its second component is always the one that has an output prefix) and w is the stack content (top is on the left). The PDA rewrite rules are directly defined from Δ as follows: $(P,Q)v_i \xrightarrow{a} (Q,P_i)w_i$ for every $P \in \mathcal{C}onst$, $Q \in \mathcal{C}onst \cup \{\mathbf{0}\}$ and every i, $i \in I$, such that $P \stackrel{\text{def}}{=} \sum_{i\in I} v_i \rhd .\overline{w_i \rhd}.P_i$.

It is now easy to see that $P_1 \parallel \overline{w_1}.Q_1 \longrightarrow^* P_2 \parallel \overline{w_2}.Q_2$ if and only if $(P_1,Q_1)w_1 \longrightarrow^* (P_2,Q_2)w_2$.

The reachability problem of extended PDA is by standard techniques reducible to reachability of ordinary PDA where at most one stack symbol is removed by performing a single transition (it is enough to replace every rule of the form $px_1x_2\ldots x_m \longrightarrow q\alpha$ by the rules $px_1 \longrightarrow p_1$, $p_1x_2 \longrightarrow p_2$, $\ldots$, $p_{m-1}x_m \longrightarrow q\alpha$ where $p_1,\ldots,p_{m-1}$ are new control states).

Since reachability of PDA is decidable (Büchi, 1964), we can conclude that reachability of ping-pong protocols with two participants is also decidable. Moreover, the reachability problem of ordinary PDA can be solved in polynomial time (Bouajjani et al., 1997; Esparza et al., 2000), which implies that reachability of ping-pong protocols with two participants is decidable also in PTIME. □

This result is tight as we know that the reachability problem for protocols with three (and more) participants is already undecidable (Hüttel and Srba, 2004).

On the other hand, the above reduction to pushdown automata can be used to decide equivalence checking with regard to strong bisimilarity, as strong bisimilarity is well known to be decidable for the

class of pushdown processes (Sénizergues, 1998). This is possible under the restriction that the label abstraction function is *local* in the sense of (Hüttel and Srba, 2004), i.e., it makes visible only constantly many outer-most (resp. inner-most) encryption keys. Nevertheless, even under this restriction secrecy is decidable. For any protocol specification P we compare it to a specification P', where P' is a version of P that has been modified as follows. We introduce a fresh key k_*. Every input prefix $w.P$ is replaced by $k_*w.P$, and similarly every output prefix $\overline{w}.P$ is replaced by $\overline{k_*w}.P$. We fix keys a and b, where $a \neq b$, and define a label abstraction function ϕ such that $\phi(k) = a$ for all $k \in \mathcal{K}$ and $\phi(w) = b$ for $w \notin \mathcal{K}$. Now ϕ is a local abstraction function, and it distinguishes between plain-texts and cipher-texts. Clearly P and P' are strongly bisimilar if and only if P never communicates a plain-text message.

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 (Focardi et al., 2000) express the environment within the process calculus, namely as a process running in parallel with the protocol. In (Amadio et al., 2002) 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

$$(C, T) \rightarrow (C', T')$$

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

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

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

$$\mathcal{S}(T) = \mathcal{A}(T) \cup \{kw \mid w \in \mathcal{S}(T),\ k \in \mathcal{K} \cap \mathcal{S}(T)\}. \tag{4}$$

The next lemmas follow immediately from Tarski's fixed-point theorem.

Lemma 2. The analysis of a set of messages $T \subseteq \mathcal{K}^*$ is the union of the family of sets $\mathcal{A}_i(S)$ defined by

$$\begin{aligned}\mathcal{A}_0(T) &= T\\ \mathcal{A}_{i+1}(T) &= \mathcal{A}_i(T) \cup \{w \mid kw \in \mathcal{A}_i(T),\ k \in \mathcal{K} \cap \mathcal{A}_i(T)\}\end{aligned}$$

Lemma 3. The synthesis of a set of messages $T \subseteq \mathcal{K}^*$ is the union of the family of sets $\mathcal{S}_i(T)$ defined by

$$\begin{aligned}\mathcal{S}_0(T) &= \mathcal{A}(T)\\ \mathcal{S}_{i+1}(T) &= \mathcal{S}_i(T) \cup \{kw \mid w \in \mathcal{S}_i(T),\ k \in \mathcal{K} \cap \mathcal{S}_i(T)\}\end{aligned}$$

The set of *compromised keys* K_c for a given set $T \subseteq \mathcal{K}^*$ of messages is defined by $K_c \stackrel{\text{def}}{=} \mathcal{K} \cap \mathcal{S}(T)$, which is easily seen to be equal to $\mathcal{K} \cap \mathcal{A}(T)$. (The compromised keys are immediately available for the intruder because they are either in his initial knowledge or can be discovered by the analysis.)

Remark 4. Let $T \subseteq \mathcal{K}^*$ be a given set of messages. The following observation is easy to verify: in order to compute the complete set K_c of compromised keys of size n, it is enough to find messages $m_1, \ldots, m_n \in T$ such that when we analyze them in a sequence, we discover exactly all compromised keys. Formally, we define

$$K_c^0 \stackrel{\text{def}}{=} \emptyset \quad \text{and} \quad K_c^i \stackrel{\text{def}}{=} K_c^{i-1} \cup \{k \in \mathcal{K} \mid m_i = wk,\ w \in (K_c^{i-1})^*\}$$

for all i, $1 \leq i \leq n$, and then $K_c = K_c^n$. □

Proposition 1. It holds that $w \in \mathcal{S}(T)$ if and only if w can be written as $w = uw'$ for some $u \in K_c^*$ and there exists $u' \in K_c^*$ such that $u'w' \in T$.

Proof. Notice that because of Lemma 2, $w \in \mathcal{A}(T)$ iff there is $u \in K_c^*$ such that $uw \in T$. The proposition then follows by an application of Lemma 3. □

We can now design an environment sensitive semantics for our calculus close in style to that of (Amadio et al., 2002). We define the reduction relation $\rightarrow$ 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).

$$(P \parallel C, T) \rightarrow (\overline{w\alpha}.P' \parallel C, T) \quad \text{if } (v\triangleright\,.\,\overline{w\triangleright}.P') \in P \text{ and } v\alpha \in \mathcal{S}(T) \qquad \text{(A1)}$$

$$(P \parallel C, T) \rightarrow (P' \parallel C, T) \quad \text{if } (v.P') \in P \text{ and } v \in \mathcal{S}(T) \qquad \text{(A2)}$$

$$(\overline{w}.P \parallel C, T) \rightarrow (P \parallel C, T \cup \{w\}) \qquad \text{(A3)}$$

$$(P \parallel C, T) \rightarrow (P' \parallel C, T \cup \{w\}) \quad \text{if } (\overline{w}.P') \in P \qquad \text{(A4)}$$

It is possible to show that this semantics can be internalized in our calculus within our existing semantics.

Theorem 4. For any recursive ping-pong protocol, we can define its new parallel component which enables all the attacks described by axioms (A1) – (A4).

The proof of this result can be found in (Hüttel and Srba, 2004). We do not include the full proof here but sketch the general construction. The necessary ideas can all be found in the proof of Lemma 1.

The attacker is an additional parallel component that keeps track of the set S of all messages that have been transmitted. S is stored in a buffer called the *message pool*, whose messages are separated using a special separator key. The message pool is encrypted using a so-called locking key. The purpose of this locking key (see also the proof of Lemma 1) is to prevent other parallel components from reading or modifying the content of the pool.

The attacker remembers the set of keys K currently known to be compromised. From this, the attacker is able to construct any message of the analysis and synthesis of S by copying any message from S to an internal buffer. The message in the internal buffer can then be altered by removing outermost keys from K and by adding keys from K. At any point, the content of the internal buffer can be transmitted to the environment.

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. This

is to be put in contrast with several results where replicative calculi are known to be capable of simulating recursive ones (see e.g. (Milner, 1993) for the case of pi-calculus which implies also the same result for spi-calculus, and (Giambiagi et al., 2004; Nielsen et al., 2002) for other examples). On the other hand, a similar discrimination result as ours between recursion and replication was recently proved for CCS (Busi et al., 2003).

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
- $\epsilon \parallel C \equiv C \equiv \ \overline{\epsilon} \parallel C$
- $!(\epsilon) \equiv \mathbf{0} \equiv !(\overline{\epsilon})$
- $!(C) \equiv C \parallel !(C)$.

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).

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

Example 3. An initial configuration $C \stackrel{\text{def}}{=} !(\overline{k}) \ \parallel !(k\triangleright \,.\, \overline{kk\triangleright})$ generates a labelled transition system in Figure 2 with infinitely many reachable configurations (only a finite fragment is depicted). Observe that (unlike recursive protocols) the number of parallel components can grow arbitrarily (e.g. the left-most branch in the picture). The reason is that we allow prefixes of the form $!(\overline{w})$ which can unconditionally output new messages and that we have replicative agents accepting these messages.

In case that the number of output prefixes for all reachable configurations is bounded by some number n, the parallel components of the form $!(v)$, $!(\overline{w})$ and $!(v \triangleright \,.\, \overline{w\triangleright})$ can be replaced by n parallel

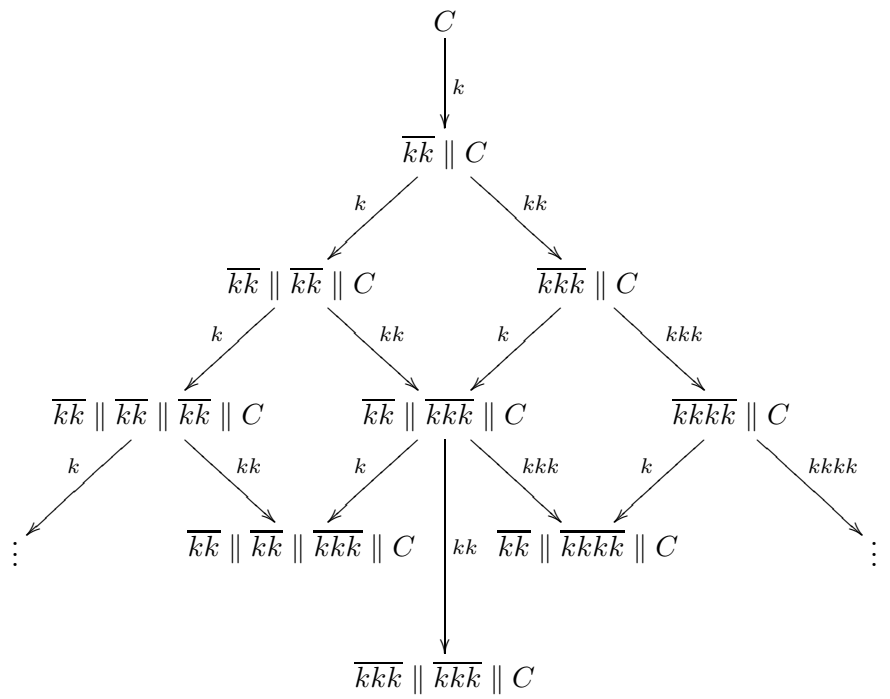

Figure 2. Initial fragment of a labelled transition system for $C \stackrel{\text{def}}{=} !(\overline{k}) \parallel !(k\triangleright\,.\,\overline{kk\triangleright})$

occurrences of fresh process constants $P_{!(v)}$, $P_{!(\overline{w})}$ and $P_{!(v\triangleright.\,\overline{w\triangleright})}$ respectively such that $P_{!(v)} \stackrel{\text{def}}{=} v.P_{!(v)}$, $P_{!(\overline{w})} \stackrel{\text{def}}{=} \overline{w}.P_{!(\overline{w})}$ and $P_{!(v\triangleright.\,\overline{w\triangleright})} \stackrel{\text{def}}{=} v\triangleright\,.\,\overline{w\triangleright}.P_{!(v\triangleright.\,\overline{w\triangleright})}$, and hence it can be simulated by recursive protocols. □

We shall 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 (Křetínský et al., 2004).

Following the work of Mayr (Mayr, 2000), let $\mathcal{C}onst$ be a set of *process constants*. The set $\mathcal{E}$ of *process expressions* over $\mathcal{C}onst$ is defined by

$$E ::= \epsilon \mid X \mid E.E \mid E \parallel E$$

where ϵ is the symbol for empty expression, X ranges over $\mathcal{C}onst$, '.' is the operator of sequential composition and '$\parallel$' is the operator of parallel composition. We identify processes up to structural congruence, which is the least congruence such that '.' is associative, '$\parallel$' is associative and commutative and ϵ is a unit for '.' and '$\parallel$'.

Let Q be a finite set of *control-states* and $\mathcal{A}ct$ a set of *actions*. A *state-extended process rewrite system* (sePRS) is a finite set Δ of rewrite

rules of the form $pE \xrightarrow{a} qF$ where $p, q \in Q$, $a \in \mathcal{A}ct$, and $E, F \in \mathcal{E}$ such that $E \neq \epsilon$.

A given sePRS Δ generates a labelled transition system $T(\Delta)$ where states are pairs of control-states and process expressions over $\mathcal{C}onst$ modulo the structural congruence, the set of actions is $\mathcal{A}ct$ and the transition relation is given by the following SOS rules (recall that '$\|$' is commutative).

$$\frac{(pE \xrightarrow{a} qF) \in \Delta}{pE \xrightarrow{a} qF} \qquad \frac{pE \xrightarrow{a} qE'}{p(E.F) \xrightarrow{a} q(E'.F)} \qquad \frac{pE \xrightarrow{a} qE'}{p(E \parallel F) \xrightarrow{a} q(E' \parallel F)}$$

A sePRS Δ is called a *weak process rewrite system* (wPRS) iff there is a partial ordering $\leq$ on Q such that all rewrite rules $pE \xrightarrow{a} qF$ from Δ satisfy that $q \leq p$.

Theorem 5. ((Křetínský et al., 2004*))* The reachability problem for wPRS is decidable.

Let us now consider an arbitrary configuration C in the calculus of replicative ping-pong protocols. We shall construct a wPRS system Δ which preserves the reachability property.

The configuration C can be naturally written as $C \equiv A \parallel B \parallel O$ where A contains all parallel components of the form $!(v\triangleright\,.\,\overline{w\triangleright})$, $!(v)$ and $!(\overline{w})$; B contains all parallel components of the form $v\triangleright\,.\,\overline{w\triangleright}$ and v; and O contains all output prefixes of the form $\overline{w}$. Here we assume that the rules of the structural congruence $\equiv$ are applied as long as possible in order to minimize the size of the configuration C (such assumptions are implicit also later on). Observe now that any configuration $C' \equiv A \parallel B' \parallel O'$ reachable from C contains exactly the same part A and every parallel component from B' is also in B.

The intuition of the reduction is that A does not have to be remembered as all parallel components from A are always available, B will be stored in control-states of the wPRS (note that there are only finitely many different components in B' for all reachable configurations of the form $A \parallel B' \parallel O'$) and the parallel components from O will be stored as a parallel composition of stacks in the wPRS system.

Let $C \equiv A \parallel B \parallel O$ be the initial configuration. Formally, the wPRS rules Δ where $\mathcal{C}onst \stackrel{\text{def}}{=} \mathcal{K} \cup \{Z, X\}$ (Z is a special symbol for the bottom of a stack, X is a process constant for creating more parallel components) and where $Q \stackrel{\text{def}}{=} \{p^{B'} \mid \exists B'' \text{ s.t. } B \equiv B' \parallel B''\}$ are defined as follows.

1.	$p^{B'}X \longrightarrow p^{B'}(X \parallel w.Z)$	$B' \subseteq B$ and $!(\overline{w}) \in A$
2.	$p^{B'}w.Z \longrightarrow p^{B'}$	$B' \subseteq B$ and $!(\overline{w}) \in A$
3.	$p^{B'}Z \longrightarrow p^{B'}$	$B' \subseteq B$
4.	$p^{B'}v.Z \longrightarrow p^{B'}$	$B' \subseteq B$ and $!(v) \in A$
5.	$p^{B'}v \longrightarrow p^{B'}w$	$B' \subseteq B$ and $!(v\triangleright . \overline{w\triangleright}) \in A$
6.	$p^{B'}v.Z \longrightarrow p^{B''}$	$B' \subseteq B$ and $B' \equiv B'' \parallel v$
7.	$p^{B'}v \longrightarrow p^{B''}w$	$B' \subseteq B$ and $B' \equiv B'' \parallel v\triangleright . \overline{w\triangleright}$

In the definitions above, whenever we have $w \in \mathcal{K}^*$, we use the word w also in the wPRS rules in the meaning that it represents a sequential composition of process constants contained in w, i.e., if $w = a_1a_2 \cdots a_n$ then w in the wPRS rules stands for the sequential composition $a_1.a_2.\ldots.a_n$. Moreover $B' \subseteq B$ means that there is some B'' such that $B \equiv B' \parallel B''$ and $x \in A$ means that the expression x is a parallel component in A. All actions are omitted as they are irrelevant for the reachability question.

Rules 1. – 3. correspond to the structural congruence $\equiv$. As $!(\overline{w}) \equiv \overline{w} \parallel !(\overline{w})$ rule 1. enables us to create a new parallel component $\overline{w}$ whenever $!(\overline{w}) \in A$ and by rule 2. such a component can always be deleted. Rule 3. corresponds to the fact that $\epsilon \parallel C \equiv C$. (Recall that we assume that C does not contain any component of the form $!(\epsilon)$ or $!(\overline{\epsilon})$.)

Rules 4. – 7. are computational rules: in rules 4. and 5. we allow the reception of messages by the components in A and the control-state does not change in this case; in rules 6. and 7. we do the same for components in B' (the current remaining part of B) and whenever such a component is used, we remove it from B' by changing the control-state to $p^{B''}$.

Let us assume the initial configuration $C \equiv A \parallel B \parallel O$ as above such that $O \equiv \overline{w_1} \parallel \overline{w_2} \parallel \cdots \parallel \overline{w_n}$. The initial configuration of the wPRS system Δ is then

$$p^B(X \parallel w_1.Z \parallel w_2.Z \parallel \cdots \parallel w_n.Z).$$

It is easy to see that every rewriting step in Δ corresponds either to a single computational step in the replicative ping-pong protocol

or to an application of some congruence rule. On the other hand, any communication in the protocol can be directly simulated in Δ.

Hence we can make the following observation (assuming that $O' \equiv \overline{w'_1} \parallel \overline{w'_2} \parallel \cdots \parallel \overline{w'_{n'}}$).

Lemma 4. It holds that

$$A \parallel B \parallel O \longrightarrow^* A \parallel B' \parallel O'$$

if and only if

$$p^B(X \parallel w_1.Z \parallel w_2.Z \parallel \cdots \parallel w_n.Z) \longrightarrow^*$$

$$p^{B'}(X \parallel w'_1.Z \parallel w'_2.Z \parallel \cdots \parallel w'_{n'}.Z).$$

Theorem 6. The reachability problem for replicative ping-pong protocols is decidable.

Proof. By Lemma 4 we reduced the problem to the reachability question for wPRS (observe that $p^{B'} \geq p^{B''}$ iff $B'' \subseteq B'$ is the natural ordering on control-states of the wPRS demonstrating that the control-state unit has a monotone behaviour). The decidability result then follows from Theorem 5. □

An interesting question to investigate is whether the active intruder (as considered in Section 4) can be explicitly modelled in the replicative variant of the ping-pong calculus. Such an intruder will have to be able to remember all the messages exchanged during the protocol execution plus he should be able to perform a set of non-trivial operations (in order to perform e.g. analysis and synthesis). We claim that the model would become Turing powerful and hence a non-trivial extension of the replicative calculus is needed to capture the behaviour of the active attacker, which will make reachability undecidable.

6. Conclusion

We have seen that ping-pong protocols extended with recursive definitions have a 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. In case that only two parties participate in the protocol, the reachability analysis becomes feasible and the answer can be provided in polynomial time.

We have also showed that reachability analysis for a replicative variant of the protocol becomes decidable. Our proof uses very recent results from process algebra (Křetínský et al., 2004) and can be compared to the work of Amadio, Lugiez and Vanackère (Amadio et al., 2002) 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 (Amadio et al., 2002) where all processes have to be replicated). In order to establish more specific decidability results for a semantics with an explicit intruder, one should investigate a setting where the most general intruder has been internalized into our calculus. The results in our paper indicate that one should probably consider a subset of the calculus that we have studied.

Last but not least we hope that our approach may be extended to include other operations on messages. That this may be possible is due to the fact that the proof of decidability for replicative protocols only uses parallel composition of stacks and consequently does not require the full generality of wPRS. 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 $k_1.(k_2 \parallel k_3).k_4$).

Other open problems include decidability of bisimilarity for replicative ping-pong protocols and the problem of determining general conditions that guarantee equi-expressiveness of recursion and replication (see e.g. (Milner, 1993; Giambiagi et al., 2004; Nielsen et al., 2002)).

Acknowledgments. We would like to thank the anonymous referees for their comments and suggestions.

References

Abadi, M. and A. Gordon: 1998, 'A Bisimulation Method for Cryptographic Protocols'. *Nordic Journal of Computing* **5**(4), 267–303.

Amadio, R. and W. Charatonik: 2002, 'On Name Generation and Set-Based Analysis in the Dolev-Yao Model'. In: *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR'02)*, Vol. 2421 of *LNCS*. pp. 499–514.

Amadio, R. and D. Lugiez: 2000, 'On the Reachability Problem in Cryptographic Protocols'. In: *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, Vol. 1877 of *LNCS*. pp. 380–394.

Amadio, R., D. Lugiez, and V. Vanackère: 2002, 'On the symbolic reduction of processes with cryptographic functions'. *Theoretical Computer Science* **290**(1), 695–740.

Boreale, M.: 2001, 'Symbolic Trace Analysis of Cryptographic Protocols'. In: *Proceedings of the 28th Colloquium on Automata, Languages and Programming (ICALP'01)*, Vol. 2076 of *LNCS*. pp. 667–681.

Bouajjani, A., J. Esparza, and O. Maler: 1997, 'Reachability Analysis of Pushdown Automata: Application to Model-Checking'. In: *Proceedings of the 8th International Conference on Concurrency Theory (CONCUR'97)*, Vol. 1243 of *LNCS*. pp. 135–150.

Büchi, J.: 1964, 'Regular canonical systems'. *Arch. Math. Logik u. Grundlagenforschung* **6**, 91–111.

Burkart, O., D. Caucal, F. Moller, and B. Steffen: 2001, 'Verification on Infinite Structures'. In: J. Bergstra, A. Ponse, and S. Smolka (eds.): *Handbook of Process Algebra*. Elsevier Science, Chapt. 9, pp. 545–623.

Busi, N., M. Gabbrielli, and G. Zavattaro: 2003, 'Replication vs. Recursive Definitions in Channel Based Calculi'. In: *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP'03)*, Vol. 2719 of *LNCS*. pp. 133–144.

Dolev, D., S. Even, and R. Karp: 1982, 'On the Security of Ping-Pong Protocols'. *Information and Control* **55**(1–3), 57–68.

Dolev, D. and A. Yao: 1983, 'On the Security of Public Key Protocols'. *Transactions on Information Theory* **IT-29**(2), 198–208.

Durgin, N., P. Lincoln, J. Mitchell, and A. Scedrov: 1999, 'Undecidability of Bounded Security Protocols'. In: N. Heintze and E. Clarke (eds.): *Proceedings of Workshop on Formal Methods and Security Protocols (FMSP'99)*.

Esparza, J., D. Hansel, P. Rossmanith, and S. Schwoon: 2000, 'Efficient algorithms for model checking pushdown systems'. In: *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'00)*, Vol. 1855 of *LNCS*. pp. 232–247.

Fiore, M. and M. Abadi: 2001, 'Computing symbolic models for verifying cryptographic protocols'. In: *14th IEEE Computer Security Foundations Workshop (CSFW '01)*. Washington - Brussels - Tokyo, pp. 160–173.

Focardi, R., R. Gorrieri, and F. Martinelli: 2000, 'Non Interference for the Analysis of Cryptographic Protocols'. In: *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP'00)*, Vol. 1853 of *LNCS*. pp. 354–372.

Giambiagi, P., G. Schneider, and F. Valencia: 2004, 'On the Expressiveness of Infinite Behavior and Name Scoping in Process Calculi'. In: *Proceedings of the 7nd International Conference on Foundations of Software Science and Computation Structures (FOSSACS'04)*, Vol. 2987 of *LNCS*. pp. 226–240.

Hüttel, H. and J. Srba: 2004, 'Recursion vs. Replication in Simple Cryptographic Protocols'. Technical Report RS-04-23, BRICS Research Series.

Hüttel, H. and J. Srba: 2004, 'Recursive Ping-Pong Protocols'. In: *Proceedings of the 4th International Workshop on Issues in the Theory of Security (WITS'04)*. pp. 129–140.

Hüttel, H. and J. Srba: 2005, 'Recursion vs. Replication in Simple Cryptographic Protocols'. In: *Proceedings of the 31st Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'05)*, Vol. 3381 of *LNCS*. pp. 175–184.

Křetínský, M., V. Řehák, and J. Strejček: 2004, 'Extended Process Rewrite Systems: Expressiveness and Reachability'. In: *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR'04)*, Vol. 3170 of *LNCS*. pp. 355–370.

Mayr, R.: 2000, 'Process Rewrite Systems'. *Information and Computation* **156(1)**, 264–286.

Milner, R.: 1993, 'The polyadic pi-calculus: a tutorial'. In: F. L. Bauer, W. Brauer, and H. Schwichtenberg (eds.): *Logic and Algebra of Specification*. Springer-Verlag, pp. 203–246.

Nielsen, M., C. Palamidessi, and F. Valencia: 2002, 'On the Expressive Power of Temporal Concurrent Constraint Programming Languages'. In: *Proceedings of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*. pp. 156–167.

Rusinowitch, M. and M. Turuani: 2003, 'Protocol Insecurity with a Finite Number of Sessions and Composed Keys is NP-complete'. *TCS: Theoretical Computer Science* **299**.

Sénizergues, G.: 1998, 'Decidability of Bisimulation Equivalence for Equational Graphs of Finite Out-Degree'. In: *Proceedings of the 39th Annual Symposium on Foundations of Computer Science(FOCS'98)*. pp. 120–129.

van Glabbeek, R.: 2001, 'The Linear Time - Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes'. In: J. Bergstra, A. Ponse, and S. Smolka (eds.): *Handbook of Process Algebra*. Elsevier Science, Chapt. 1, pp. 3–99.