BRICS

Basic Research in Computer Science

BRICS RS-03-47 Hüttel & Srba: Recursive Ping-Pong Protocols

Recursive Ping-Pong Protocols

Hans Hüttel
Jiří Srba

BRICS Report Series RS-03-47

ISSN 0909-0878 December 2003

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/03/47/`

Recursive Ping-Pong Protocols

Hans Hüttel* and Jiří Srba**

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

Abstract. This paper introduces a process calculus with recursion which allows us to express an unbounded number of runs of the ping-pong protocols introduced by Dolev and Yao. We study the decidability issues associated with two common approaches to checking security properties, namely reachability analysis and bisimulation checking. Our main result is that our channel-free and memory-less calculus is Turing powerful, assuming that at least three principals are involved. We also investigate the expressive power of the calculus in the case of two participants. Here, our main results are that reachability and, under certain conditions, also strong bisimilarity become decidable.

1 Introduction

The study of correctness properties of cryptographic protocols has become an increasingly important research topic. Today, research on cryptographic protocols is often conducted using methods from program semantics together with the so-called Dolev-Yao assumptions about protocol principals and intruders introduced in [11]. In the Dolev-Yao model, all communications of a protocol may be visible to the hostile environment which is capable of interfering with the protocol by altering or blocking any message and by creating new messages. Moreover, these are the only kinds of attacks — an intruder cannot exploit weaknesses of the encryption algorithm itself (the 'perfect encryption hypothesis').

Process calculi have been suggested as a natural vehicle for reasoning about cryptographic protocols. In [1], 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 observational equivalence. Alternatively, security properties can be expressed and examined using reachability analysis [4, 6, 14]. An important question is: Given the Dolev-Yao assumptions, to which extent are the properties of cryptographic protocols decidable?

A number of security properties are decidable for the class of finite protocols [4, 16]. In the case of an unbounded number of protocol configurations, the picture is more complex. Durgin et al. showed in [12] that security properties are

[$\star$] hans@cs.auc.dk

[$\star\star$] srba@cs.auc.dk, supported in part by the GACR, grant No. 201/03/1161.

[$\star\star\star$] **B**asic **R**esearch in **C**omputer **S**cience, Centre of the Danish National Research Foundation.

undecidable in a restricted class of so-called bounded protocols (that still allows for infinitely many reachable configurations). In [3] 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 [5] that the reachability problem is in PTIME for a class of protocols with replication (as opposed to recursion).

Another contribution by Dolev and Yao in [11] is the study of *ping-pong protocols*. These are 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 [10] by expressing secrecy as emptiness of the intersection of a context-free language with a regular language. The class of protocols studied by Amadio et al. in [5] contains iterative ping-pong protocols and, as a consequence, secrecy properties remain polynomially decidable even in this case.

In this paper we examine decision problems for a process calculus capable of describing exactly the class of *recursive* ping-pong protocols. We study the calculus from the perspective of equivalence checking (for a general scheme see e.g. [15]) and consider no explicit model of the environment. The reason for considering this tiny calculus is that all negative results for it carry over to richer calculi capable of expressing a wider class of protocols (it is possible to describe an active intruder in the setting of bisimilarity/reachability checking of ping-pong protocols and this will be treated in our forthcoming paper).

The considered calculus is a channel-free process calculus, reminiscent of the tail-recursive processes studied by Amadio and Charatonik [3]. In the case of three principals, we can encode any Turing machine as a ping-pong protocol. Hence even this very restricted formalism is sufficiently expressive to encode universal computations. This implies that any richer calculus (and indeed any reasonable cryptographic calculus should subsume the ping-pong behaviour) is beyond the reach of automatic verification. The underlying idea of our construction is to encode the content of the tape as a series of encryptions and to express the transition function as a series of protocol steps. As a consequence, both reachability and bisimilarity are undecidable.

On the other hand, if the protocol is restricted to two principals, many properties become decidable. In particular, we show that the reachability problem is in PTIME, and under a certain natural observational condition also strong bisimilarity becomes decidable.

2 Basic Definitions

2.1 Transition Systems and Bisimilarity

We describe the semantics of our process calculi using unlabelled transition systems where every state has an associated set of its *knowledge*, which is an element of a given domain $\mathcal{D}$. Intuitively, knowledge represents observations visible to the environment.

A *transition system (with knowledge)* is a triple $(S, \longrightarrow, kn)$ where S is a set of *states* (or *processes*), $\longrightarrow \subseteq S \times S$ is a *transition relation*, written $\alpha \longrightarrow \beta$, for $(\alpha, \beta) \in \longrightarrow$, and $kn : S \mapsto \mathcal{D}$ is a *knowledge function* from the set of states to the given knowledge domain $\mathcal{D}$.

Let $\mathcal{T} = (S, \longrightarrow, kn)$ be a transition system. A binary relation $R \subseteq S \times S$ is a *(strong) bisimulation* iff whenever $(\alpha, \beta) \in R$ then $kn(\alpha) = kn(\beta)$ and if $\alpha \longrightarrow \alpha'$ then $\beta \longrightarrow \beta'$ for some β' such that $(\alpha', \beta') \in R$, and if $\beta \longrightarrow \beta'$ then $\alpha \longrightarrow \alpha'$ for some α' such that $(\alpha', \beta') \in R$.

Processes $\alpha_1, \alpha_2 \in S$ are *(strongly) bisimilar* in $\mathcal{T}$, written $(\alpha_1, \mathcal{T}) \sim (\alpha_2, \mathcal{T})$ (or simply $\alpha_1 \sim \alpha_2$ if $\mathcal{T}$ is clear from the context), iff there is a (strong) bisimulation R such that $(\alpha_1, \alpha_2) \in R$.

Given a pair of processes α_1 in a transition system $\mathcal{T}_1 = (S_1, \longrightarrow_1, kn)$ and α_2 in $\mathcal{T}_2 = (S_2, \longrightarrow_2, kn)$ such that $S_1 \cap S_2 = \emptyset$ and $kn : S_1 \cup S_2 \mapsto \mathcal{D}$, we write $(\alpha_1, \mathcal{T}_1) \sim (\alpha_2, \mathcal{T}_2)$ iff $(\alpha_1, \mathcal{T}) \sim (\alpha_2, \mathcal{T})$ such that $\mathcal{T} \overset{\text{def}}{=} (S_1 \cup S_2, \longrightarrow, kn)$ where $\alpha \longrightarrow \beta$ iff $\alpha, \beta \in S_1$ and $\alpha \longrightarrow_1 \beta$, or $\alpha, \beta \in S_2$ and $\alpha \longrightarrow_2 \beta$. Similarly if S_1 and S_2 are not disjoint, we simply rename the states of one of the transition systems and use the same notation $(\alpha_1, \mathcal{T}_1) \sim (\alpha_2, \mathcal{T}_2)$.

Bisimilarity has an elegant characterization in terms of *bisimulation games* [19, 18]. A bisimulation game on a pair of processes $(\alpha_1, \mathcal{T})$ and $(\alpha_2, \mathcal{T})$ is a two-player game of an 'attacker' and a 'defender'. The game is played in rounds. In each round the attacker chooses one of the processes and performs a transition in the selected process; the defender must respond by performing a transition in the other process. Now the game repeats, starting from the new processes. If a pair of processes α_1 and α_2 such that $kn(\alpha_1) \neq kn(\alpha_2)$ is reached during the game, the attacker wins. If a player cannot perform a transition, the other player wins. If the game is infinite, the defender wins.

Processes $(\alpha_1, \mathcal{T})$ and $(\alpha_2, \mathcal{T})$ are bisimilar iff the defender has a winning strategy (and non-bisimilar iff the attacker has a winning strategy).

Two main decidability problems we shall investigate are (strong) bisimilarity checking and reachability analysis. The first problem asks the question (i) Are two given states α and β in a transition system (strongly) bisimilar ($\alpha \sim \beta$) ? and the second problem asks the question (ii) Is a given state β reachable from a state α, i.e., $\alpha \longrightarrow^* \beta$?

2.2 Ping-Pong Protocols

Let T be a set of *plain-text* messages and let K be a set of *encryption keys.* We let t range over T and k range over K. The set of messages over T encrypted with the keys from K is denoted by $\mathcal{M}(T, K)$ and given by the following abstract syntax.

$$m ::= \quad t \quad | \quad \{m\}_k$$

Hence a message m (either a plain-text or an already encrypted message) can be encrypted with a key k, and such a message is written as $\{m\}_k$.

Let $\mathcal{C}onst$ be a finite set of *process constants*. A *specification of a ping-pong protocol* is a finite set of *process definitions* Δ such that every process constant $P \in \mathcal{C}onst$ has exactly one process definition of the form

$$P \stackrel{\mathrm{def}}{=} \sum_{i \in I} v_i.\overline{w_i}.P_i$$

where I is a finite set of indices, v_i and w_i are messages over a fixed variable name x encrypted with the keys from K (i.e. $v_i, w_i \in \mathcal{M}(\{x\}, K)$), and P_i is either a process constant or the *empty process* '$\mathbf{0}$' (i.e. $P_i \in \mathcal{C}onst \cup \{\mathbf{0}\}$). We call v_i (resp. $\overline{w_i}$) the *input* (resp. *output*) prefix.

Process definitions like these will often be written in their unfolded forms $P \stackrel{\mathrm{def}}{=} v_1.\overline{w_1}.P_1 + v_2.\overline{w_2}.P_2 + \cdots + v_n.\overline{w_n}.P_n$ and whenever P_i is the empty process $\mathbf{0}$ then instead of $v_i.\overline{w_i}.\mathbf{0}$ we write only $v_i.\overline{w_i}$. Also, let $keys(u)$ denote the set of keys used in u for $u \in \mathcal{M}(T, K)$, formally $keys(t) \stackrel{\mathrm{def}}{=} \emptyset$ and $keys(\{m\}_k) \stackrel{\mathrm{def}}{=} \{k\} \cup keys(m)$.

Example 1. A simple protocol specification may look as follows:

$$\Delta \stackrel{\mathrm{def}}{=} \{P \stackrel{\mathrm{def}}{=} \{x\}_k.\overline{\{\{x\}_{k'}\}_k}.P\}$$

where $K \stackrel{\mathrm{def}}{=} \{k, k'\}$. The cyclic behaviour of the process constant P is described as follows: P receives a message and decrypts it by using the key k; the decrypted message is encrypted (using first the key k' and then k), and sent off.

Let us finally also recall that for the prefixes $\{x\}_k$ and $\{\{x\}_{k'}\}_k$ we have $keys(\{x\}_k) = \{k\}$ and $keys(\{\{x\}_{k'}\}_k) = \{k, k'\}$. □

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

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

where $P \in \mathcal{C}onst \cup \{\mathbf{0}\}$ ranges over process constants including the empty process, $w \in \mathcal{M}(T, K)$ ranges over the set of messages, and '$\|$' is the operator of the parallel composition.

We introduce a structural congruence which identifies configurations that represent the same state of the protocol. $\equiv$ 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. We identify configurations up to structural congruence.

The *width* of a configuration C is the minimal number of the parallel components in the $\equiv$-equivalence class represented by C. Hence e.g. the width of $\mathbf{0}$ is 0 and the width of $P \| \mathbf{0} \| Q$ is 2.

We shall now impose two natural restrictions on knowledge functions. We say that a knowledge function $kn : \mathcal{C}onf \mapsto \mathcal{D}$ for a given set $\mathcal{D}$ *respects structural equivalence* if $C_1 \equiv C_2$ implies $kn(C_1) = kn(C_2)$ for any $C_1, C_2 \in \mathcal{C}onf$. Let $C \in \mathcal{C}onf$ be a configuration with the corresponding specification Δ. The set $prefix(C)$

of available prefixes of C is defined by: $\mathit{prefix}(\mathbf{0}) \overset{\text{def}}{=} \emptyset$; $\mathit{prefix}(C) \overset{\text{def}}{=} \cup_{i\in I}\{v_i, \overline{w_i}\}$ if $C \in \mathcal{C}onst$ such that $\big(C \overset{\text{def}}{=} \sum_{i\in I} v_i.\overline{w_i}.P_i\big) \in \Delta$; $\mathit{prefix}(C) \overset{\text{def}}{=} \{\overline{w}\}$ if $C = \overline{w}.P$ for some $P \in \mathcal{C}onst$; and $\mathit{prefix}(C) \overset{\text{def}}{=} \mathit{prefix}(C_1) \cup \mathit{prefix}(C_2)$ if $C = C_1\|C_2$. The intuition is that the input/output capabilities of a configuration depend only on the available prefixes and not on the names of process constants. If we consistently rename the process constants in C and Δ and obtain a new configuration C' and a new specification Δ', it is the case that $\mathit{prefix}(C) = \mathit{prefix}(C')$. Hence we say that a knowledge function *kn respects renaming of process constants* if $\mathit{prefix}(C_1) = \mathit{prefix}(C_2)$ implies $kn(C_1) = kn(C_2)$ for any $C_1, C_2 \in \mathcal{C}onf$.

We only consider knowledge functions that respect structural congruence and renaming of process constants; we will call such functions *respecting.*

Example 2. Define the following knowledge functions $kn_\emptyset$, kn_{priv} and kn_{plain}:

- The *empty knowledge function* $kn_\emptyset : \mathcal{C}onf \mapsto \{\bot\}$ is defined by $kn_\emptyset(C) \overset{\text{def}}{=} \bot$ for all $C \in \mathcal{C}onf$.
- The *private-keys knowledge function* $kn_{priv} : \mathcal{C}onf \mapsto 2^K$ (where K is the set of keys) is defined by $kn_{priv}(C) \overset{\text{def}}{=} \bigcup_{i\in I} \mathit{keys}(v_i)$ if $C \in \mathcal{C}onst$ and $C \overset{\text{def}}{=} \sum_{i\in I} v_i.\overline{w_i}.P_i$, $kn_{priv}(C) \overset{\text{def}}{=} kn_{priv}(C_1) \cup kn_{priv}(C_2)$ if $C = C_1\|C_2$ and $kn_{priv}(C) \overset{\text{def}}{=} \emptyset$ otherwise.
- The *plain-text knowledge function* $kn_{plain} : \mathcal{C}onf \mapsto 2^T$ (where T is the set of plain-text messages), is defined by $kn_{plain}(C) \overset{\text{def}}{=} \{t\}$ if $C = \overline{t}.P$ for some $t \in T$ and $P \in \mathcal{C}onst$, $kn_{plain}(C) \overset{\text{def}}{=} kn_{plain}(C_1) \cup kn_{plain}(C_2)$ if $C = C_1\|C_2$ and $kn_{plain}(C) \overset{\text{def}}{=} \emptyset$ otherwise.

It is easy to see that these knowledge functions are respecting. The intuition is that $kn_\emptyset$ does not take the knowledge of configurations into account and hence in bisimilarity checking only the branching structure induced by $\longrightarrow$ is relevant. The knowledge function kn_{priv} makes sure that any two bisimilar configurations have the same decryption keys currently available. Finally, the function kn_{plain} identifies configurations which are currently capable of communicating the same set of plain-text messages. □

Example 3. Consider a ping-pong protocol motivated by a simple example mentioned e.g. in [10] and [11]. A participant X wants to send a message $m \in \{0,1\}^*$ to a participant Y and get a confirmation that the message was received. The protocol is informally described as follows: (i) participant X encrypts the message m by a public key of Y and sends the encrypted message to Y, (ii) participant Y decrypts the received message by his private key and answers to X by the same message m encrypted with X's public key, (iii) finally X receives the confirmation message and decrypts it by his private key.

In our formalism let $T \overset{\text{def}}{=} \{0,1\}^*$, $K \overset{\text{def}}{=} \{k_X, k_Y\}$, and $\mathcal{C}onst \overset{\text{def}}{=} \{X, Y\}$. The protocol specification Δ is given by two equations

$$X \overset{\text{def}}{=} \{x\}_{k_X}.\overline{\{x\}_{k_Y}}.X \qquad Y \overset{\text{def}}{=} \{x\}_{k_Y}.\overline{\{x\}_{k_X}}$$

$$\frac{(P \overset{\mathrm{def}}{=} \sum_{i\in I} v_i.\overline{w_i}.P_i) \in \mathit{\Delta} \qquad u = v_i[m/x] \qquad m \in \mathcal{M}(T,K) \qquad i \in I}{P \parallel \overline{u}.Q \longrightarrow \overline{w_i[m/x]}.P_i \parallel Q}$$

$$\frac{P \longrightarrow P'}{P \| Q \longrightarrow P' \| Q}$$

Fig. 1. SOS rules of ping-pong protocols

and the initial configuration of the protocol is $\overline{\{m\}_{k_Y}}.X \parallel Y$ for a given $m \in T$.

The intuition is that the process $\overline{\{m\}_{k_Y}}.X$ can output the message m encrypted with the key k_Y and become the process X. Similarly, Y can input this message and become the process $\overline{\{m\}_{k_X}}.Y$. After the communication is established, the new configuration $X \parallel \overline{\{m\}_{k_X}}.Y$ is reached. The conformation phase of the protocol is analogous to the first communication. □

Note that we do not distinguish explicitly between private and public keys. This is done implicitly: a key supposed to be a private key for one or more process constants cannot be used in the input prefixes of other process constants. E.g. in our example k_X is a private key of X which means the process constant Y can use the key only in its output prefix and vice versa.

A formal semantics of ping-pong protocols is given in terms of transition systems. First, we define inductively a substitution $u[m'/x]$ of the message m' for the variable x in the prefix u ($m' \in \mathcal{M}(T,K)$ and $u \in \mathcal{M}(\{x\},K)$) by $x[m'/x] \overset{\mathrm{def}}{=} m'$ and $\{m\}_k[m'/x] \overset{\mathrm{def}}{=} \{m[m'/x]\}_k$.

A given protocol specification $\mathit{\Delta}$ determines a transition system $\mathcal{T}(\mathit{\Delta}) \overset{\mathrm{def}}{=} (S, \longrightarrow, \mathit{kn})$ where states are configurations of the protocol modulo the structural congruence ($S \overset{\mathrm{def}}{=} \mathcal{C}\mathit{onf}/{\equiv}$); the transition relation $\longrightarrow$ is given by the SOS rules in Figure 1 (recall that '$\|$' is commutative); and the knowledge function $\mathit{kn} : S \mapsto \mathcal{D}$ is assumed to be explicitly given. Note that the width of a configuration does not increase by performing a transition.

Example 4. Let us consider the ping-pong protocol specification from Example 3. The interesting fragment of the transition system $\mathcal{T}(\mathit{\Delta})$ is

$$\overline{\{m\}_{k_Y}}.X \parallel Y \xrightarrow{\qquad} X \parallel \overline{\{m\}_{k_X}}.Y \xrightarrow{\qquad} \overline{\{m\}_{k_Y}}.X \qquad □$$

In the rest of this section we will discuss the usefulness of bisimilarity checking with knowledge functions for validation of authenticity and secrecy of ping-pong protocols. The correctness checking of such protocols is done by comparing the protocol specification with its ideal behaviour [2] under an appropriate choice of the knowledge function.

For example, assume that we want to check whether a given protocol specification $\mathit{\Delta}$ ever outputs a plain-text message. Let us fix an arbitrary key $k \in K$.

We define an ideal protocol Δ' which has the same behaviour as Δ but never communicates any plain-text message. This can be achieved e.g. by defining

$$\Delta' \overset{\text{def}}{=} \{P \overset{\text{def}}{=} \sum_{i\in I} \{v_i\}_k.\overline{\{w_i\}_k}.P_i \mid (P \overset{\text{def}}{=} \sum_{i\in I} v_i.\overline{w_i}.P_i) \in \Delta\}.$$

For any configuration $C \in \mathcal{C}onf$ let C' be a configuration where every occurrence of an output prefix of the form $\overline{w}.P$ is replaced with $\overline{\{w\}_k}.P$. Under the assumption that the plain-text knowledge function kn_{plain} is used, it holds that $(C, \mathcal{T}(\Delta)) \sim (C', \mathcal{T}(\Delta'))$ if and only if the protocol Δ starting from its initial configuration C is never capable of outputting any plain-text message.

Assume now that Δ contains input prefixes only of the from $\{x\}_k$ for some key $k \in K$ (only one key decryption at a time). The question whether a computation from a given configuration C of the protocol specification Δ never deadlocks and it is always possible to decrypt messages encrypted with a fixed key k can be expressed as follows. Let $\Delta' \overset{\text{def}}{=} \{X \overset{\text{def}}{=} \{x\}_k.\overline{\{x\}_k}.X\}$ and let C' be the configuration $\overline{\{t\}_k}.X \parallel X$ for some $t \in T$. Obviously, from C' there is exactly one transition leading back to C' and moreover C' is always capable of decrypting a message encrypted with the key k. We also define a knowledge function $kn : \mathcal{C}onf \mapsto \{0, 1\}$ by $kn(C_1) \overset{\text{def}}{=} 1$ if $k \in kn_{priv}(C_1)$; and $kn(C_1) \overset{\text{def}}{=} 0$ otherwise. Here kn_{priv} is the private-keys knowledge function introduced before. The considered validation question is now equivalent to the problem $(C, \mathcal{T}(\Delta)) \sim (C', \mathcal{T}(\Delta'))$.

Remark 1. If instead of kn defined above we use $kn_\emptyset$, the bisimilarity question $(C, \mathcal{T}(\Delta)) \sim (C', \mathcal{T}(\Delta'))$ is equivalent to the problem whether there is no terminating computation of the protocol Δ starting in C.

3 Ping-Pong Protocols of Width 3

In this section we show that ping-pong protocols of width at least 3 are surprisingly powerful enough to simulate Turing machines.

Let $M = (Q, \Sigma, \gamma, q_0, q_F)$ be a *Turing machine* such that Q is a finite set of *control states*, Σ is a finite *tape alphabet* containing special symbols $¢, \$ \in \Sigma$ ($¢$ is the left mark of the tape and \$ it the right mark; let $\Sigma_1 \overset{\text{def}}{=} \Sigma \smallsetminus \{¢, \$\}$), $q_0 \in Q$ is the *initial state*, q_F is the *final state* and

$$\gamma : \Sigma \times Q \times \Sigma \mapsto (Q \times \Sigma \times \Sigma \;\cup\; \Sigma \times \Sigma \times Q \;\cup\; \{halt\})$$

is a total function such that

- $\gamma(aqb) \in \big(Q \times \{a\} \times \Sigma_1 \;\cup\; \{a\} \times \Sigma_1 \times Q\big)$ for all $a, b \in \Sigma_1$ and $q \in Q$ (the head moves either to the left or to the right)
- $\gamma(¢qa) \in \{¢\} \times \Sigma_1 \times Q$ for all $a \in \Sigma$ and $q \in Q$ (the head is not allowed to move on the left mark)
- $\gamma(aq\$) \in \big(Q \times \{a\} \times \{\$\} \;\cup\; \{a\} \times \Sigma_1 \times Q\big)$ for all $a \in \Sigma$ and $q \in Q$ (the right mark cannot be changed but the head can move to the right)

- $\gamma(aq_Fb) = \mathit{halt}$ for all $a, b \in \Sigma$
  (when the final state q_F is reached, the computation stops).

A *configuration* of the machine M is an element from the set $\{¢\}\times\Sigma_1^*\times Q\times\Sigma_1^*\times\{\$\}$. A *computational step* between configurations c_1 and c_2 (written $c_1 \longrightarrow c_2$) is defined in the usual way, i.e.,

- $c_1 \longrightarrow c_2$ if $c_1 \equiv w_1aqbw_2$ and $c_2 \equiv w_1\gamma(aqb)w_2$ where $w_1, w_2 \in \Sigma^*$, $a, b \in \Sigma$ and $q \in Q$ such that $b \neq \$$, or $\big(b = \$ \ \wedge \ \gamma(aqb) \in Q\times\{a\}\times\{\$\}\big)$ (the head does not write on the right mark), or
- $c_1 \longrightarrow c_2$ if $c_1 \equiv w_1aq\$$ and $c_2 \equiv w_1\gamma(aq\$)\$$ where $w_1 \in \Sigma^*$, $a \in \Sigma$ and $q \in Q$ such that $\gamma(aq\$) \in \{a\}\times\Sigma_1\times Q$ (the head is at the end mark and moves to the right).

It is a well known fact that the problem whether the machine *halts* from the initial configuration $¢q_0\$$ (i.e. reaches a configuration containing the state q_F in a finite number of computational steps) is undecidable.

Let $M = (Q, \Sigma, \gamma, q_0, q_F)$ be a Turing machine and let $S \overset{\text{def}}{=} Q \cup \Sigma \cup \{z\}$ and $S' \overset{\text{def}}{=} \{s' \mid s \in S\}$ such that z is a fresh symbol and $S \cap S' = \emptyset$. We define a ping-pong protocol Δ that will simulate the computation of the machine M. The set of plain-text messages is a singleton set $T \overset{\text{def}}{=} \{t\}$, the set of keys is $K \overset{\text{def}}{=} S \cup S'$, and the set of process constants consists of $\mathcal{C}onst \overset{\text{def}}{=} \{B_S, B_{S'}, P, R\} \cup \{P_{a'}, R_a \mid a \in S\} \cup \{V_{a'b'c'}, V_{a'b'c'\$'} \mid a, b, c \in S\}$.

First, we define the equations for process constants B_S and $B_{S'}$ (buffers over S and S').

$$B_S \overset{\text{def}}{=} \sum_{s\in S} \{x\}_s.\overline{\{x\}_s}.B_S \qquad\qquad B_{S'} \overset{\text{def}}{=} \sum_{s'\in S'} \{x\}_{s'}.\overline{\{x\}_{s'}}.B_{S'}$$

The intuition is that the buffers B_S and B'_S can store their content as a sequence of encryption keys over S resp. S'. In our case the buffers will store configurations of the machine M.

The process constant P transfers the content of the buffer B_S to the buffer $B_{S'}$ and as soon as a control state is present, the computational step is performed. This is formally defined by

$$\begin{aligned} P \overset{\text{def}}{=} & \sum_{a,b,c\in S,\ b\notin Q} \{\{\{x\}_c\}_b\}_a.\overline{\{\{x\}_c\}_b}.P_{a'} \ + \\ & \sum_{a,c\in\Sigma,\ q\in Q,\ \gamma(aqc)=efg,\ \neg\omega} \{\{\{x\}_c\}_q\}_a.\overline{x}.V_{e'f'g'} \ + \\ & \sum_{a,c\in\Sigma,\ q\in Q,\ \gamma(aqc)=efg,\ \omega} \{\{\{x\}_c\}_q\}_a.\overline{x}.V_{e'f'g'\$'} \end{aligned}$$

where ω is the condition saying that the head is at the end of the tape and it moves to the right, i.e., $\omega \ \equiv \ \ c = \$ \ \wedge \ g \in Q$.

The process constant $P_{a'}$ for all $a' \in S'$ simply adds the symbol a' to the buffer $B_{S'}$ and then it continues as P.

$$P_{a'} \overset{\text{def}}{=} \sum_{s' \in S'} \{x\}_{s'}.\overline{\{\{x\}_{s'}\}_{a'}}.P$$

The process constants $V_{a'b'c'}$ and $V_{a'b'c'\$'}$ (for $a', b', c' \in S'$) add the corresponding sequence of keys to the buffer $B_{S'}$ and then continue as R.

$$V_{a'b'c'} \overset{\text{def}}{=} \sum_{s' \in S'} \{x\}_{s'}.\overline{\{\{\{\{x\}_{s'}\}_{a'}\}_{b'}\}_{c'}}.R$$

$$V_{a'b'c'\$'} \overset{\text{def}}{=} \sum_{s' \in S'} \{x\}_{s'}.\overline{\{\{\{\{\{x\}_{s'}\}_{a'}\}_{b'}\}_{c'}\}_{\$'}}.R$$

We finish the definition of the process constants by introducing R (standing for 'reverse'), which transfers the content of the buffer $B_{S'}$ back to B_S.

$$R \overset{\text{def}}{=} \sum_{a' \in S'} \{x\}_{a'}.\overline{x}.R_a$$

Similarly as $P_{a'}$, R_a for all $a \in S \smallsetminus \{¢\}$ adds the symbol a to the buffer B_S and continues as R, except for the situation when the beginning of the tape was reached (in this case $R_{¢}$ continues as P).

$$R_a \overset{\text{def}}{=} \sum_{s \in S} \{x\}_s.\overline{\{\{x\}_s\}_a}.R \qquad R_{¢} \overset{\text{def}}{=} \sum_{s \in S} \{x\}_s.\overline{\{\{x\}_s\}_{¢}}.P$$

Let $m \overset{\text{def}}{=} \{t\}_z$ and $m' \overset{\text{def}}{=} \{t\}_{z'}$ (recall that $t \in T$ is the only plain-text message). The following configuration of width 3 can simulate the computation of the machine M from the initial configuration $¢q_0\$$.

$$\overline{\{\{\{m\}_{\$}\}_{q_0}\}_{¢}}.B_S \;\|\; \overline{m'}.B_{S'} \;\|\; P$$

In order to see how the simulation works we use the notations $[w]_m$ and $[w]_{m'}$ to denote the messages m and m' encrypted with the sequence of keys w, i.e., $[\epsilon]_m \overset{\text{def}}{=} m$, $[\epsilon]_{m'} \overset{\text{def}}{=} m'$, $[aw]_m \overset{\text{def}}{=} \{[w]_m\}_a$ and $[aw]_{m'} \overset{\text{def}}{=} \{[w]_{m'}\}_a$ where ϵ is the empty sequence, $a \in K$ and $w \in K^*$.

Now every configuration c of the machine M corresponds to the configuration $f(c) \overset{\text{def}}{=} (\overline{[c]_m}.B_S \parallel \overline{[\epsilon]_{m'}}.B_{S'} \parallel P)$ of the protocol Δ. Note that the initial configuration of Δ defined above is exactly $f(¢q_0\$)$.

The following considerations will describe the simulation of the Turing machine M by the protocol Δ. A single step of the machine M will be simulated by a finite number of transitions in the protocol.

Let $c_1 = w_1aqcw_2$ and $c_2 = w_1efgw_2$ be configurations of M such that $w_1, w_2 \in \Sigma^*$, $a, c \in \Sigma$, $q \in Q$, $\gamma(aqc) = efg$, and $\neg\omega$. This means that $c_1 \longrightarrow c_2$. We will show that $f(c_1) \longrightarrow^* f(c_2)$ such that the computation of the protocol from $f(c_1)$ is deterministic, i.e., for any $C \in \mathcal{C}onf$ such that $f(c_1) \longrightarrow^* C$ it is the case that $C \longrightarrow C'$ and $C \longrightarrow C''$ implies $C' \equiv C''$. For w_1 in the form $a_1 \cdots a_n$ let $w'_{1,R} \overset{\text{def}}{=} a'_n \cdots a'_1$ be the reversed word w_1 such that every letter is primed. The computation from $f(c_1)$ looks as follows.

$$f(c_1) = \overline{[w_1aqcw_2]_m}.B_S \parallel \overline{[\epsilon]_{m'}}.B_{S'} \parallel P \longrightarrow^* \overline{[aqcw_2]_m}.B_S \parallel \overline{[w'_{1,R}]_{m'}}.B_{S'} \parallel P \longrightarrow$$

$$B_S \parallel \overline{[w'_{1,R}]_{m'}}.B_{S'} \parallel \overline{[w_2]_m}.V_{e'f'g'} \longrightarrow \overline{[w_2]_m}.B_S \parallel \overline{[w'_{1,R}]_{m'}}.B_{S'} \parallel V_{e'f'g'} \longrightarrow$$

$$\overline{[w_2]_m}.B_S \parallel B_{S'} \parallel \overline{[g'f'e'w'_{1,R}]_{m'}}.R \longrightarrow \overline{[w_2]_m}.B_S \parallel \overline{[g'f'e'w'_{1,R}]_{m'}}.B_{S'} \parallel R \longrightarrow^*$$

$$\overline{[w_1efgw_2]_m}.B_S \parallel \overline{[\epsilon]_{m'}}.B_{S'} \parallel P = f(c_2)$$

It is easy to observe that such a computation is unique (deterministic).

Let $c_1 = w_1aq\$$ and $c_2 = w_1efg\$$ be configurations of M as before, however, this time the condition ω holds. This case is analogous to the previous one. The only difference is that $V_{e'f'g'}$ is replaced with $V_{e'f'g'\$'}$ and the end of the tape $\$$ is added (the tape hence becomes longer by one cell). Assume now that $f(c_1)$ represents a halting configuration. The computation of the protocol from $f(c_1)$ starts as before, however, there is no summand in the definition of the process constant P for the situation that $\gamma(aqc) = \mathit{halt}$ and hence the computation gets stuck in the configuration $\overline{[aqcw_2]_m}.B_S \parallel \overline{[w'_{1,R}]_{m'}}.B_{S'} \parallel P$.

The following theorems are applications of the presented simulation.

Theorem 1. *Reachability is undecidable for ping-pong protocols of width* 3*.*

Theorem 2. *Bisimilarity is undecidable for ping-pong protocols of width* 3 *for any knowledge function which respects structural congruence and renaming of process constants.*

4 Ping-Pong Protocols of Width 2

If the family of protocols we consider contains at most two participants (parallel components), we get a class similar to that of the traditional ping-pong protocols [10, 11]. 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 $w \in \mathcal{M}(T, K)$. If this is not the case, the computation of such a protocol is stuck — no communication can take place.

We shall now observe that the class of ping-pong protocols of width 2 is not Turing powerful since e.g. reachability becomes decidable. Hence we can still hope for automatic verification of some protocol properties.

Theorem 3. *The reachability problem for ping-pong protocols of width* 2 *is decidable in polynomial time.*

In contrast, the bisimilarity problem is again undecidable.

Theorem 4. *The problem of bisimilarity checking between a pair of ping-pong protocol configurations of width* 2 *is undecidable (provided that we allow for general but still computable, respecting and finite-domain knowledge functions).*

In most of the examples of knowledge functions we, however, do not use knowledge functions similar to the one described in the undecidability proof. In fact, it is quite satisfactory to consider knowledge functions which depend only on a constant number of the upper-most symbols in the output prefix. We shall prove that if this is the case then bisimilarity becomes decidable.

Let $P \parallel \overline{w}.Q$ be a general form of a protocol configuration of width 2. A knowledge function *kn* is *local* if there is a constant $M \in \mathbb{N}^0$ and a function

$$f : (\mathcal{C}onst \cup \{\mathbf{0}\}) \times (\mathcal{C}onst \cup \{\mathbf{0}\}) \times \big((K^{<M} \times T) \cup K^M\big) \mapsto \mathcal{D}$$

such that $kn(P \parallel \overline{w}.Q) = f(P, Q, w')$ where w' is $\langle w \rangle$ if $|\langle w \rangle| \leq M$, and w' is the prefix of $\langle w \rangle$ of length M otherwise. In other words, a local knowledge function depends only on P, Q and at most M outer-most keys of w.

Remark 2. Observe that local knowledge functions have finite co-domains. Hence for every local knowledge function there is an equivalent local knowledge function with a finite domain $\mathcal{D}$. In what follows we shall assume that $\mathcal{D}$ is finite.

Theorem 5. *The problem of bisimilarity checking between a pair of ping-pong protocol configurations of width* 2 *is decidable for local knowledge functions.*

5 Conclusion

We have studied a simple, channel-free process calculus capable of describing recursive ping-pong protocols. Surprisingly, the calculus turns out to be Turing-powerful when we allow three or more principals. This implies that all interesting verification problems will remain undecidable in any richer calculus which can express at least the ping-pong behaviour. This fact remains valid even if we allow active attacks on the protocol since the syntax of ping-pong protocols is capable of describing this situation. This is, however, nontrivial to see and it is a part of our current work.

In the case of two principals, the reachability problem is in PTIME. Depending on our notion of observability, other properties (including strong bisimilarity) may also become decidable.

Amadio et al. in [5] have proved that reachability is polynomially decidable for processes with replication. However, they have only shown the result for a class of processes that, unlike the class studied in the present paper, does not involve an explicit representation of nondeterministic choice. It remains to be seen whether security properties are decidable for a version of our process calculus with replication replacing recursion, and whether our calculus without explicit nondeterminism remains Turing powerful. We claim that at least the latter is indeed the case and in our future work we shall further investigate the problem (including the connection with the results from [9] where it is shown that secrecy is decidable for protocols with replication but without nondeterminism).

References

1. Martin Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.
2. Martin Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.
3. R.M. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR'02)*, volume 2421 of *LNCS*, pages 499–514. Springer-Verlag, 2002.
4. R.M. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *LNCS*, pages 380–394. Springer-Verlag, 2000.
5. Roberto M. Amadio, Denis Lugiez, and Vincent Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, October 2002.
6. Michele Boreale. Symbolic trace analysis of cryptographic protocols. In *28th Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of *LNCS*, pages 667–681. Springer, July 2001.
7. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proceedings of the 8th International Conference on Concurrency Theory (CONCUR'97)*, volume 1243 of *LNCS*, pages 135–150. Springer-Verlag, 1997.
8. J.R. Büchi. Regular canonical systems. *Arch. Math. Logik u. Grundlagenforschung*, 6:91–111, 1964.
9. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proceedings of Rewriting Techniques and Applications (RTA'03)*, number 2706 in LNCS, pages 148–164. Springer-Verlag, 2003.
10. D. Dolev, S. Even, and R.M. Karp. On the security of ping-pong protocols. *Information and Control*, 55(1–3):57–68, 1982.
11. D. Dolev and A.C. Yao. On the security of public key protocols. *Transactions on Information Theory*, IT-29(2):198–208, 1983.
12. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In N. Heintze and E. Clarke, editors, *Proceedings of Workshop on Formal Methods and Security Protocols (FMSP'99)*, July 1999.
13. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'00)*, volume 1855 of *LNCS*, pages 232–247. Springer-Verlag, 2000.
14. M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *14th IEEE Computer Security Foundations Workshop (CSFW '01)*, pages 160–173, Washington - Brussels - Tokyo, June 2001. IEEE.
15. R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 354–372. Springer-Verlag, 2000.
16. M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete. *TCS: Theoretical Computer Science*, 299, 2003.

17. G. Sénizergues. 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)*, pages 120–129. IEEE Computer Society, 1998.
18. C. Stirling. Local model checking games. In *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995.
19. W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science (extended abstract). In *Proceedings of the 4th International Joint Conference CAAP/FASE, Theory and Practice of Software Development (TAPSOFT'93)*, volume 668 of *LNCS*, pages 559–568. Springer-Verlag, 1993.

A Appendix

Theorem 1. *Reachability is undecidable for ping-pong protocols of width* 3.

Proof. Let M be a Turing machine and let Δ be the ping-pong protocol constructed above. We modify Δ by adding the following summand to the definition of the process constant P

$$\sum_{a,c\in\Sigma,\ q\in Q,\ \gamma(aqc)=halt} \{\{\{x\}_c\}_q\}_a.\overline{x}.D$$

where D is a new process constant with its definition

$$D \overset{\text{def}}{=} \sum_{a\in S\cup S'} \{x\}_a.\overline{x}.D.$$

This means that M halts if and only if a protocol configuration containing D is reachable. The process constant D can remove the content of both of the buffers and hence the question whether the configuration $\overline{m}.B_S \parallel \overline{m'}.B_{S'} \parallel D$ is reachable from the initial configuration $f(¢q_0\$)$ is undecidable. □

Theorem 2. *Bisimilarity is undecidable for ping-pong protocols of width* 3 *for any knowledge function which respects structural congruence and renaming of process constants.*

Proof. Let M be a Turing machine and let Δ be the ping-pong protocol constructed above. Let Δ' be a copy of Δ such that every process constant $X \in \mathcal{C}onst$ in Δ' is replaced with X'. Moreover, Δ also contains the following extra summand in the definition of the process constant P

$$\sum_{a,c\in\Sigma,\ q\in Q,\ \gamma(aqc)=halt} \{\{\{x\}_c\}_q\}_a.\overline{\{\{\{x\}_c\}_q\}_a}$$

and Δ' contains the following extra summand in the definition of the process constant P' plus a definition of a fresh process constant Z'

$$\sum_{a,c\in\Sigma,\ q\in Q,\ \gamma(aqc)=halt} \{\{\{x\}_c\}_q\}_a.\overline{\{\{\{x\}_c\}_q\}_a}.Z'$$

$$Z' \overset{\text{def}}{=} \sum_{a,c\in\Sigma,\ q\in Q,\ \gamma(aqc)=halt} \{\{\{x\}_c\}_q\}_a.\overline{\{\{\{x\}_c\}_q\}_a}.Z'.$$

If the machine M diverges then the initial configurations (as described above) of Δ and Δ' are bisimilar since both of them are capable of performing infinite computations (these computations are deterministic) and the knowledge function cannot distinguish between them (it respects renaming of process constants).

If the machine M halts then Δ' can still perform an infinite sequence of transitions but Δ gets stuck after using the extra summand in P defined above. Hence their initial configurations cannot be bisimilar for any knowledge function. □

Theorem 3. *The reachability problem for ping-pong protocols of width* 2 *is decidable in polynomial time.*

Proof. We reduce reachability of ping-pong protocols of width 2 to reachability of pushdown automata (PDA), disregarding the input alphabet. 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 $p_1\alpha_1 \longrightarrow^* p_2\alpha_2$.

Let $m \in \mathcal{M}(T,K)$. By $\langle m\rangle \in (K \cup T)^*$ we understand the sequence of keys that occur in m followed by the corresponding plain-text message, i.e., $\langle t\rangle = t$ for $t \in T$ and $\langle\{m\}_k\rangle = k\langle m\rangle$. Similarly, if $m \in \mathcal{M}(\{x\},K)$ then $\langle m\rangle \in K^*$ denotes the sequence of keys that occur in m, i.e., $\langle x\rangle = \epsilon$ and $\langle\{m\}_k\rangle = k\langle m\rangle$.

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 contains the encryption keys plus plain-text messages, i.e., it is the set $K \cup T$. The set of (input) actions is a singleton set $\{a\}$. 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 $(P,Q)\langle w\rangle$ such that (P,Q) is the control state (its second component is always the one that has an output prefix) and $\langle w\rangle$ is the stack content. The PDA rewrite rules are defined as follows: $(P,Q)\langle v_i\rangle \xrightarrow{a} (Q,P_i)\langle w_i\rangle$ for every $P \in \mathcal{C}onst$ and $Q \in \mathcal{C}onst \cup \{\mathbf{0}\}$ such that $P \overset{\text{def}}{=} \sum_{i\in I} v_i.\overline{w_i}.P_i$.

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 $(P_1,Q_1)\langle w_1\rangle \longrightarrow^* (P_2,Q_2)\langle w_2\rangle$.

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 $px_1 \longrightarrow p_1$, $p_1x_2 \longrightarrow p_2$, …, $p_{m-1}x_m \longrightarrow q\alpha$ where $p_1,\ldots,p_{m-1}$ are new control states).

Since reachability of PDA is decidable [8], we can conclude that reachability of ping-pong protocols of width 2 is also decidable. Moreover, the reachability problem of ordinary PDA can be solved in polynomial time [7, 13], which implies that reachability of ping-pong protocols of width 2 is decidable also in PTIME. □

Definition 1. *A* Minsky machine R *with two counters* c_1 *and* c_2 *is a finite sequence of* instructions

$$R = (I_1,\quad I_2,\quad \ldots,\quad I_{n-1},\quad n : \texttt{halt})$$

where $n \geq 1$ *and every* I_p, $1 \leq p \leq n-1$ *is an instruction of one from the following two types:*

- *increment:* $p:\ c_i := c_i + 1;$ `goto` q
- *test and decrement:* $p:$ `if` $c_i = 0$ `then goto` q `else` $c_i := c_i - 1;$ `goto` r

where $1 \leq i \leq 2$ *and* $1 \leq q, r \leq n$.

Definition 2. *A* configuration *of a Minsky machine* R *is a triple* (p, v_1, v_2) *where* p *is an instruction label* ($1 \leq p \leq n$), *and* $v_1, v_2 \in \mathbb{N}^0$ *are nonnegative integers representing the values of the counters* c_1 *and* c_2, *respectively. The transition relation* $\longrightarrow$ *between configurations is defined in the natural way.*

Note that the computation of R is deterministic, i.e., if $(p, v_1, v_2) \longrightarrow (p', v_1', v_2')$ and $(p, v_1, v_2) \longrightarrow (p'', v_1'', v_2'')$ the $p' = p''$, $v_1' = v_1''$ and $v_2' = v_2''$.

Definition 3. *A Minsky machine* R halts *with the initial counter values set to zero if* $(1, 0, 0) \longrightarrow^* (n, v_1, v_2)$ *for some* $v_1, v_2 \in \mathbb{N}^0$. *If* R *does not halt we say that it* diverges.

Proposition 1. *The halting problem for Minsky machines in undecidable.*

Let R be a given Minsky machine. We now construct a ping-pong protocol specification Δ and two configurations C_1 and C_2 of width 2 such that R diverges if and only if $(C_1, \mathcal{T}(\Delta)) \sim (C_2, \mathcal{T}(\Delta))$.

Let the set of plain-text messages be $T \overset{\text{def}}{=} \{t\}$, let the set of encryption keys be $K \overset{\text{def}}{=} \{1, \ldots, n\} \cup \{+_i, -_i \mid 1 \leq i \leq 2\} \cup \{p^i_{=0}, p^i_{\geq 0}, p^{cheat} \mid 1 \leq p < n, 1 \leq i \leq 2\}$ and let $\mathit{Const} \overset{\text{def}}{=} \{B_K, P, Q\}$. The constant B_K stands for a buffer over a certain subset of K.

Let $\#_k(m)$ denote the number of occurrences of the key $k \in K$ in the message $m \in \mathcal{M}(T, K)$. The intuition of the reduction is that a configuration (p, v_1, v_2) of the Minsky machine R corresponds to a pair of protocol configurations $\overline{\{m\}_p}.B_K \parallel P$ and $\overline{\{m\}_p}.B_K \parallel Q$ such that $\#_{+_i}(m) - \#_{-_i}(m) = v_i$ for $1 \leq i \leq 2$. The definitions of the process constants P and Q are almost symmetric except for the situation when a halting configuration of the machine R is reachable. In this case the computation from the configuration containing Q is stuck while the configuration containing P performs an infinite sequence of transition. The knowledge function is designed in such a way that if a correct computation of the machine R is simulated by the attacker in the bisimulation game (a single step in the computation of R will be simulated by two transitions in Δ), the defender can only mimic the same transitions in the other process. However, if the attacker "cheats" in the bisimulation game (e.g. decreases a value of a counter into negative integers), the defender threatens by entering a syntactically equal (and hence bisimilar) pair of protocol configurations.

Formally, the protocol Δ is given as follows.

$$B_K \overset{\text{def}}{=} \sum_{k \in \{1, \ldots, n\}} \{x\}_k.\overline{\{x\}_k}.B_K \; +$$

$$\sum_{All(p)} \Big(\{x\}_{p^i_{=0}}.\overline{\{x\}_p}.B_K \; + \; \{x\}_{p^i_{\geq 0}}.\overline{\{x\}_p}.B_K \; +$$

$$\{x\}_{p^{cheat}}.\overline{\{x\}_p}.B_K \Big)$$

$$
\begin{gathered}
P \stackrel{\text{def}}{=} \sum_{Inc(p)} \{x\}_p.\overline{\{\{x\}_{+_i}\}_q}.P \; + \\
\sum_{Dec(p)} \Big(\{x\}_p.\overline{\{x\}_{q^i_{=0}}}.P \;+\; \{x\}_p.\overline{\{x\}_{q^{cheat}}}.P \; + \\
\{x\}_p.\overline{\{x\}_{q^{cheat}}}.Q \Big) \; + \\
\sum_{Dec(p)} \Big(\{x\}_p.\overline{\{\{x\}_{-_i}\}_{r^i_{\geq 0}}}.P \;+\; \{x\}_p.\overline{\{\{x\}_{-_i}\}_{r^{cheat}}}.P \; + \\
\{x\}_p.\overline{\{\{x\}_{-_i}\}_{r^{cheat}}}.Q \Big) \;+\; \{x\}_n.\overline{\{x\}_n}.P
\end{gathered}
$$

$$
\begin{gathered}
Q \stackrel{\text{def}}{=} \sum_{Inc(p)} \{x\}_p.\overline{\{\{x\}_{+_i}\}_q}.Q \; + \\
\sum_{Dec(p)} \Big(\{x\}_p.\overline{\{x\}_{q^i_{=0}}}.Q \;+\; \{x\}_p.\overline{\{x\}_{q^{cheat}}}.Q \; + \\
\{x\}_p.\overline{\{x\}_{q^{cheat}}}.P \Big) \; + \\
\sum_{Dec(p)} \Big(\{x\}_p.\overline{\{\{x\}_{-_i}\}_{r^i_{\geq 0}}}.Q \;+\; \{x\}_p.\overline{\{\{x\}_{-_i}\}_{r^{cheat}}}.Q \; + \\
\{x\}_p.\overline{\{\{x\}_{-_i}\}_{r^{cheat}}}.P \Big)
\end{gathered}
$$

where $Inc(p) \stackrel{\text{def}}{=} 1 \leq p < n \;\wedge\; I_p = (p: \;\; c_i := c_i + 1\texttt{; goto } q)$, and $Dec(p) \stackrel{\text{def}}{=} 1 \leq p < n \;\wedge\; I_p = (p: \texttt{if } c_i = 0 \texttt{ then goto } q \texttt{ else } c_i := c_i - 1\texttt{; goto } r)$, and $All(p) \stackrel{\text{def}}{=} 1 \leq p \leq n \;\wedge\; 1 \leq i \leq 2$.

We shall now argue that R diverges if and only if

$$\overline{\{t\}_1}.B_K \parallel P \quad \sim \quad \overline{\{t\}_1}.B_K \parallel Q$$

in $\mathcal{T}(\Delta)$ where the knowledge function

$$kn : \mathcal{C}onf \mapsto \{OK_{=0}, OK_{\geq 0}, CHEAT, OTHER\}$$

is defined as follows (let $i \in \{1,2\}$, $X, Y \in \{B_K, P, Q\}$, and $p \in \{1, \ldots, n-1\}$).

$$kn(X \parallel \overline{\{m\}_{p^i_{=0}}}.Y) \stackrel{\text{def}}{=} \begin{cases} OK_{=0} & \text{if } \#_{+_i}(m) = \#_{-_i}(m) \\ CHEAT & \text{otherwise} \end{cases}$$

$$kn(X \parallel \overline{\{m\}_{p^i_{\geq 0}}}.Y) \overset{\text{def}}{=} \begin{cases} OK_{\geq 0} & \text{if } \#_{+_i}(m) \geq \#_{-_i}(m) \\ CHEAT & \text{otherwise} \end{cases}$$

$$kn(X \parallel \overline{\{m\}_{p^{cheat}}}.Y) \overset{\text{def}}{=} CHEAT$$

For all other configurations C let $kn(C) \overset{\text{def}}{=} OTHER$.

Remark 3. Note that kn is a computable and respecting knowledge function, and that its knowledge domain is finite.

Lemma 1. *If R halts then the attacker has a winning strategy in the bisimulation game played from the pair of configurations $\overline{\{t\}_1}.B_K \parallel P$ and $\overline{\{t\}_1}.B_K \parallel Q$.*

Proof. Assume that $(1, 0, 0) \longrightarrow^* (n, v_1, v_2)$ for some $v_1, v_2 \in \mathbb{N}^0$. We will show that the attacker can force the defender to reach a pair of configurations

$$\overline{\{m\}_n}.B_K \parallel P \quad \text{and} \quad \overline{\{m\}_n}.B_K \parallel Q$$

for some $m \in \mathcal{M}(T, \{+_i, -_i \mid 1 \leq i \leq 2\})$ such that $\#_{+_i}(m) - \#_{-_i}(m) = v_i$ for $1 \leq i \leq 2$. From this pair of configurations the attacker wins because

$$\overline{\{m\}_n}.B_K \parallel P \longrightarrow B_K \parallel \overline{\{m\}_n}.P$$

whereas $\overline{\{m\}_n}.B_K \parallel Q \not\longrightarrow$.

In order to show that the attacker can force the defender to faithfully simulate the computation of the Minsky machine, we assume that the current configuration of R is (p, v_1, v_2) where $1 \leq p < n$ and that the bisimulation game starts from the pair

$$\overline{\{m\}_p}.B_K \parallel P \quad \text{and} \quad \overline{\{m\}_p}.B_K \parallel Q$$

such that $\#_{+_i}(m) - \#_{-_i}(m) = v_i$ for $1 \leq i \leq 2$. We will show that if $(p, v_1, v_2) \longrightarrow (p', v'_1, v'_2)$ then after two rounds of the bisimulation game the attacker can force the defender to reach a pair of configurations $\overline{\{m'\}_{p'}}.B_K \parallel P$ and $\overline{\{m'\}_{p'}}.B_K \parallel Q$ such that $\#_{+_i}(m') - \#_{-_i}(m') = v'_i$ for $1 \leq i \leq 2$. There are three situations to be discussed.

(1) If the instruction I_p is of the form

$$p: \;\; c_i := c_i + 1; \;\texttt{goto}\; q$$

then the attacker makes two moves

$$\overline{\{m\}_p}.B_K \parallel P \longrightarrow B_K \parallel \overline{\{\{m\}_{+_i}\}_q}.P \longrightarrow \overline{\{\{m\}_{+_i}\}_q}.B_K \parallel P$$

and the defender can only answer by

$$\overline{\{m\}_p}.B_K \parallel Q \longrightarrow B_K \parallel \overline{\{\{m\}_{+_i}\}_q}.Q \longrightarrow \overline{\{\{m\}_{+_i}\}_q}.B_K \parallel Q.$$

(2) If the instruction I_p is of the form

$$p: \;\texttt{if}\; c_i = 0 \;\texttt{then goto}\; q \;\texttt{else}\; c_i := c_i - 1; \;\texttt{goto}\; r$$

and $v_i = 0$ then the attacker plays

$$\overline{\{m\}_p}.B_K \parallel P \longrightarrow B_K \parallel \overline{\{m\}_{q^i_{=0}}}.P \longrightarrow \overline{\{m\}_q}.B_K \parallel P.$$

This time the defender has six choices how to respond to the first move of the attacker. However, notice that

$$kn(B_K \parallel \overline{\{m\}_{q^i_{=0}}}.P) = OK_{=0}$$

because $v_i = 0$ and hence the defender can only answer by

$$\overline{\{m\}_p}.B_K \parallel Q \longrightarrow B_K \parallel \overline{\{m\}_{q^i_{=0}}}.Q \longrightarrow \overline{\{m\}_q}.B_K \parallel Q.$$

In all other possible moves the defender loses after the first move because the knowledge function returns different values.

(3) If the instruction I_p is of the form

$$p : \texttt{ if } c_i = 0 \texttt{ then goto } q \texttt{ else } c_i := c_i - 1\texttt{; goto } r$$

and $v_i > 0$ then the attacker plays

$$\overline{\{m\}_p}.B_K \parallel P \longrightarrow B_K \parallel \overline{\{\{m\}_{-_i}\}_{r^i_{\geq 0}}}.P \longrightarrow \overline{\{\{m\}_{-_i}\}_r}.B_K \parallel P.$$

Again, because

$$kn(B_K \parallel \overline{\{\{m\}_{-_i}\}_{r^i_{\geq 0}}}.P) = OK_{\geq 0}$$

the defender can only answer by

$$\overline{\{m\}_p}.B_K \parallel Q \longrightarrow B_K \parallel \overline{\{\{m\}_{-_i}\}_{r^i_{\geq 0}}}.Q \longrightarrow \overline{\{\{m\}_{-_i}\}_r}.B_K \parallel Q.$$

□

Lemma 2. *If R diverges then the defender has a winning strategy in the bisimulation game played from the pair $\overline{\{t\}_1}.B_K \parallel P$ and $\overline{\{t\}_1}.B_K \parallel Q$.*

Proof. We will show that the defender in the bisimulation game from $\overline{\{t\}_1}.B_K \parallel P$ and $\overline{\{t\}_1}.B_K \parallel Q$ can force the attacker to faithfully simulate the computation of the Minsky machine. Since the computation of R diverges, the bisimulation game is infinite and the defender wins.

Consider a configuration (p, v_1, v_2) where $1 \leq p < n$ of the machine R that was reached during the computation from $(1, 0, 0)$. Let the corresponding pair of protocol configurations be

$$\overline{\{m\}_p}.B_K \parallel P \quad \text{and} \quad \overline{\{m\}_p}.B_K \parallel Q$$

such that $\#_{+_i}(m) - \#_{-_i}(m) = v_i$ for $1 \leq i \leq 2$.

We will show that if $(p, v_1, v_2) \longrightarrow (p', v_1', v_2')$ then after two rounds of the bisimulation game the defender can force the attacker to reach a pair of configurations $\overline{\{m'\}_{p'}}.B_K \parallel P$ and $\overline{\{m'\}_{p'}}.B_K \parallel Q$ such that $\#_{+_i}(m') - \#_{-_i}(m') = v_i'$

for $1 \leq i \leq 2$, or the defender can win by reaching a pair of syntactically equal (and hence bisimilar) configurations. There are three situations to be discussed.

(1) If the instruction I_p is of the form

$$p: \;\; c_i := c_i + 1\texttt{; goto } q$$

then there is only one possible continuation of the bisimulation game such that after two rounds the players reach the pair

$$\overline{\{\{m\}_{+_i}\}_q}.B_K \parallel P \;\text{ and }\; \overline{\{\{m\}_{+_i}\}_q}.B_K \parallel Q.$$

(2) If the instruction I_p is of the form

$$p: \texttt{if } c_i = 0 \texttt{ then goto } q \texttt{ else } c_i := c_i - 1\texttt{; goto } r$$

and $v_i = 0$, the attacker is forced to make either the move $\overline{\{m\}_p}.B_K \parallel P \longrightarrow B_K \parallel \overline{\{m\}_{q^i_{=0}}}.P$ or $\overline{\{m\}_p}.B_K \parallel Q \longrightarrow B_K \parallel \overline{\{m\}_{q^i_{=0}}}.Q$ and the game faithfully simulates the computation of R as before. In all other attacker's moves the defender wins:

- if the attacker takes any of the four transitions (either from P or Q) introducing the key of the form p^{cheat} as the upper most encryption, the defender simply mimics the corresponding move in the other configuration except for the fact that he may switch P for Q or vice versa in order to achieve a pair of syntactically equal configurations (the knowledge function will return $CHEAT$ in these situations);
- if the attacker takes a transition that should decrement a value of the counter c_i but the counter is already empty, i.e.,

  $$\overline{\{m\}_p}.B_K \parallel P \longrightarrow B_K \parallel \overline{\{\{m\}_{-_i}\}_{r^i_{\geq 0}}}.P \quad \text{or}$$

  $$\overline{\{m\}_p}.B_K \parallel Q \longrightarrow B_K \parallel \overline{\{\{m\}_{-_i}\}_{r^i_{\geq 0}}}.Q,$$

  the defender answers by the transitions

  $$\overline{\{m\}_p}.B_K \parallel Q \longrightarrow B_K \parallel \overline{\{\{m\}_{-_i}\}_{r^{cheat}}}.P \quad \text{resp.}$$

  $$\overline{\{m\}_p}.B_K \parallel P \longrightarrow B_K \parallel \overline{\{\{m\}_{-_i}\}_{r^{cheat}}}.Q.$$

  Since the attacker "cheated" the knowledge function allows this response (it returns the value $CHEAT$ for these configurations). After the second round (there is only one possible continuation of the game which transfers the encrypted messages to the buffer), the protocol configurations become syntactically equal and hence the attacker loses.

(3) If the instruction I_p is of the form

$$p: \texttt{if } c_i = 0 \texttt{ then goto } q \texttt{ else } c_i := c_i - 1\texttt{; goto } r$$

and $v_i > 0$, the situation is similar to the previous case. □

Theorem 4. *The problem of bisimilarity checking between a pair of ping-pong protocol configurations of width* 2 *is undecidable (provided that we allow for general but still computable, respecting and finite-domain knowledge functions).*

Proof. From Lemma 1 and Lemma 2. □

Theorem 5. *The problem of bisimilarity checking between a pair of ping-pong protocol configurations of width* 2 *is decidable for local knowledge functions.*

Proof. We shall reduce our problem to strong bisimilarity checking of PDA. The result then follows from the fact that strong bisimilarity of PDA is decidable [17].

In the reduction we extend the construction provided in the proof of Theorem 3. We consider not only the PDA rules

$$(P, Q)\langle v_i \rangle \xrightarrow{a} (Q, P_i)\langle w_i \rangle$$

for every $P \in \mathcal{C}onst$ and $Q \in \mathcal{C}onst \cup \{\mathbf{0}\}$ such that $P \overset{\text{def}}{=} \sum_{i \in I} v_i.\overline{w_i}.P_i$ but also the rules

$$(P, Q)\langle w' \rangle \overset{f(P,Q,w')}{\longrightarrow} (P, Q)\langle w' \rangle$$

for all elements (P, Q, w') in the co-domain of the function f. Hence the reduction steps in the ping-pong protocol correspond to the a-labelled transitions in the PDA system, and the condition that the knowledge function agrees on bisimilar states is tested in the PDA by executing loops labelled by the elements from $\mathcal{D}$. It is now easy to see that protocol configurations $P_1 \parallel \overline{w_1}.Q_1$ and $P_2 \parallel \overline{w_2}.Q_2$ are bisimilar if and only if the PDA configurations $(P_1, Q_1)\langle w_1 \rangle$ and $(P_2, Q_2)\langle w_2 \rangle$ are strongly bisimilar. □

Remark 4. By standard techniques for pushdown automata one can extend the previous theorem to allow a slightly more general definition of local knowledge functions. In particular, the functions *kn* can depend also on a constant number of inner-most keys of the encrypted message in addition to the M outer-most keys.

Recent BRICS Report Series Publications

RS-03-47 Hans Hüttel and Jiří Srba. *Recursive Ping-Pong Protocols*. December 2003. 21 pp. To appear in the proceedings of 2004 IFIP WG 1.7, ACM SIGPLAN and GI FoMSESS Workshop on Issues in the Theory of Security (WITS'04).

RS-03-46 Philipp Gerhardy. *The Role of Quantifier Alternations in Cut Elimination*. December 2003. 10 pp. Extends paper appearing in Baaz and Makowsky, editors, *European Association for Computer Science Logic: 17th International Workshop*, CSL '03 Proceedings, LNCS 2803, 2003, pages 212-225.

RS-03-45 Peter Bro Miltersen, Jaikumar Radhakrishnan, and Ingo Wegener. *On converting CNF to DNF*. December 2003. 11 pp. A preliminary version appeared in Rovan and Vojtás, editors, *Mathematical Foundations of Computer Science: 28th International Symposium*, MFCS '03 Proceedings, LNCS 2747, 2003, pages 612–621.

RS-03-44 Anna Gál and Peter Bro Miltersen. *The Cell Probe Complexity of Succinct Data Structures*. December 2003. 17 pp. An early version of this paper appeared in Baeten, Lenstra, Parrow and Woeginger, editors, *30th International Colloquium on Automata, Languages, and Programming*, ICALP '03 Proceedings, LNCS 2719, 2003, pages 332–344.

RS-03-43 Mikkel Nygaard and Glynn Winskel. *Domain Theory for Concurrency*. December 2003. 45 pp. To appear in a *Theoretical Computer Science* special issue on Domain Theory.

RS-03-42 Mikkel Nygaard and Glynn Winskel. *Full Abstraction for HOPLA*. December 2003. 25 pp. Appears in Amadio and Lugiez, editors, *Concurrency Theory: 14th International Conference*, CONCUR '03 Proceedings, LNCS 2761, 2003, pages 383–398.

RS-03-41 Malgorzata Biernacka, Dariusz Biernacki, and Olivier Danvy. *An Operational Foundation for Delimited Continuations*. December 2003. 21 pp.

RS-03-40 Andrzej Filinski and Henning Korsholm Rohde. *A Denotational Account of Untyped Normalization by Evaluation*. December 2003. 29 pp.