Towards Automatic Bisimilarity Checking in the Spi Calculus

Anders Strandløv Elkjær Michael Höhle
Hans Hüttel* Kasper Overgård Nielsen
{ase,hoehle,hans,kaspero}@@cs.auc.dk

BRICS, Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E
9220 Aalborg Øst
Denmark

October 14, 1998

Abstract

The spi calculus by Abadi and Gordon, an extension of Robin Milner's π-calculus, is designed to model cryptographic protocols. Classic security properties are easily expressed in spi using the notion of testing equivalence by De Nicola and Hennessy. However, proving processes testing equivalent is a daunting task. Thus framed bisimilarity, a bisimulation method implying testing equivalence, has been proposed by Abadi and Gordon. Unfortunately this method is immediately not suited for automation, as the definition of framed bisimilarity uses several levels of quantification over infinite domains. In this paper we define *fenced bisimilarity*, a concept similar to framed bisimilarity in which one of these quantifiers has been eliminated. We prove that fenced bisimilarity is a sound and complete characterization of framed bisimilarity. Though fenced bisimilarity is a step towards automation, there is still work to be done before a fully automated tool can be implemented.

1 Introduction

The boom of the Internet as a major carrier for electronic exchange of information and the extended use of distributed systems, such as cellular phones, has made cryptography an indispensable element to restrict access to information and avoid fraud. A cryptographic protocol consists of 2-5 messages used to identify users and authenticate transactions. Despite their size, quite a number of protocols contain serious design blunders[AN95]. Such blunders make it easier for an intruder to attack the protocol rather than trying to break its message enciphering algorithm (e.g. IDEA [Sch96]). Several attempts on establishing formal verification methods for cryptographic protocols have been made [BAN89, AG97a, AG97b, MMS97]; some require that the behavior of the attacker has to be modelled, others operate on too high levels of abstraction thus making the relation to the actual implementation tenuous.

The spi-calculus [AG97a, AG97b] strikes a golden mean: it does not require any assumptions about the behaviour of an attacker (except that the it should be expressible within the calculus), it has a precise semantics and is directly executable. The spi-calculus is an extension of the π-calculus [Mil91], which is a calculus for modeling concurrent communicating processes. The spi-calculus is extended with cryptographic primitives that allow encrypted communication similar to the secret channel communication in the π-calculus. In the real world, truly secret channels cannot exist, hence this notion is used to describe internal communication and not communication between principals on, for example, the Internet.

*Contact author (for further correspondence)

When verifying spi-processes, comparing processes with their specifications is the main idea. But to what degree should a process equal its specification? This is where the notion of testing equivalence is suggested by Abadi and Gordon. The notion testing equivalence, first introduced by DeNicola and Hennessy, identifies two processes if they allow the same end observations, no matter what observer they interact with. In the setting of the spi calculus, the observers are potentially malicious, and consequently two processes are equivalent if they can resist the same attacks.

However, it is rarely possible to check directly if two processes are testing equivalent as the definition contains a universal quantification over attackers. Therefore it is necessary to find other, more tractable notions of equivalence which imply testing equivalence. Various notions of bisimilarity, defined by Park [Par81], are popular candidates. Strong bisimilarity was the important notion of behavioural equivalence in Milner's work on CCS[Mil89] and various versions of bisimilarity have since been defined for the π-calculus[Mil91]. A major advantage of using bisimilarity is that its very definition allows one to prove equivalence by means of a coinductive technique: if you want to show that two processes are equivalent, it is enough to exhibit a so-called bisimulation containing them.

However, the notion of strong bisimulation used in the π-calculus is too restrictive to be useful for cryptographic protocols. There are numerous examples of processes that are testing equivalent, but not strongly bisimilar. A less restrictive, but still sound, notion of bisimulation has been introduced by Abadi and Gordon in [AG98b], the notion of framed bisimilarity. It relates processes given the knowledge provided to the environment about them and has a coinductive definition.

Abadi and Gordon conjecture that framed bisimilarity can be fully automated. Normally bisimulation equivalences are well suited for automation, because checking can be done by constructing a bisimulation. Unfortunately, this is not immediately the case with framed bisimulation, as its definition uses several levels of quantification over infinite domains.

This paper presents a sound and complete characterization of framed bisimilarity, called *fenced bisimilarity*. Fenced bisimulation replaces one of these infinite existential quantifiers with a terminating algorithm and is therefore a significant step towards a fully effective characterization of framed bisimilarity.

Contents of this paper

Section 2 introduces the spi-calculus, Section 3 explains framed bisimilarity, Section 4 introduces fenced bisimilarity and Section 5 discusses the results and offers ideas for further work.

2 The spi calculus

The spi-calculus is a small process calculus used to model cryptographic protocols. Communication takes place over channels, that can either be public or restricted. Messages that are communicated can be encrypted, such that only those *possessing* the key can decrypt them. This is a basic assumption when using the spi-calculus; it is not possible for an attacker to guess the key of an encrypted message. In real life an attacker *could* guess the key or find it by crypto-analysis. However the focus of the spi-calculus is to investigate properties of protocols and not the encryption methods.

2.1 Syntax of agents

We have chosen the same basic definition of spi as Abadi and Gordon [AG97a].

2.1.1 Terms

The basic building blocks of the spi-calculus are terms, defined by the following grammar.

$$L, M, N ::= 0 \mid x \mid m \mid \mathrm{suc}(M) \mid (M, N) \mid \{M\}_N$$

Since terms can be 0 and $\mathrm{suc}(M)$, this gives us the ability to construct all natural numbers. Contrary to the π-calculus the spi-calculus distinguishes between variables (x, y, z) and names (c, m, n, k). A name refers to a key or a channel.

When concerning channels there exists a polarity between a name c used for input and its co-name $\overline{c}$ used for output. Allowing terms to be pairs of terms facilitates communicating arrays of terms, since the elements in a pair can be pairs themselves. Finally, you can encrypt ($\{M\}_N$) a term M using a key N. The result is a ciphertext that can only be decrypted if N is known. [AG97a] also contains versions of spi with public key encryption and hash functions. In this paper we only consider shared key cryptography since the definitions in [AG98b, AG98c] of indistinguishability under environments and proofs about framed bisimilarity only use shared key cryptography.

2.1.2 Processes

Processes can be introduced as programs executed in the calculus. Their syntax is defined as follows:

$$
\begin{aligned}
P, Q, R ::= &\ (\nu n)P \\
&\mid \overline{M}\langle N\rangle.P \\
&\mid M(x).P \\
&\mid P \mid Q \\
&\mid\ !P \\
&\mid [M \,\mathrm{is}\, N]P \\
&\mid \mathbf{0} \\
&\mid \mathrm{let}\ (x, y) = M \,\mathrm{in}\, P \\
&\mid \mathrm{case}\, M \,\mathrm{of}\ \ 0 : P\, \mathrm{suc}(x) : Q \\
&\mid \mathrm{case}\, L \,\mathrm{of}\ \ \{x\}_N \,\mathrm{in}\, P
\end{aligned}
$$

$(\nu n)P$ is the restriction operator; The new name n is bound in P and no process outside the scope of P knows about it, unless P at some point reveals it.

What really makes processes useful is their ability to communicate. $\overline{M}\langle N\rangle.P$ is output; N is sent out on the channel M. $M(x).P$ is input; the variable x is received on the channel M, and x is bound in P, just like the restriction operator.

$\mathrm{let}\ (x, y) = M \,\mathrm{in}\, P$ is used to split pairs; the variables x and y are bound in P.

$\mathrm{case}\, M \,\mathrm{of}\, 0 : P\, \mathrm{suc}(x) : Q$ executes P if M is 0 and $Q[N/x]$ if M is the successor of a term N; in this case x is bound in Q.

Finally, $\mathrm{case}\, L \,\mathrm{of}\, \{x\}_N \,\mathrm{in}\, P$ is used to decrypt terms; x is bound in P.

In the spi-calculus processes are identified up to renaming of bound names. We define a closed process to be a process without any free variables, and *Proc* to be the set of closed processes. Furthermore we let $\mathrm{fn}[\![P]\!]$ denote the set of free names in P, and $\mathrm{fv}[\![P]\!]$ the free variables in P.

It is important to notice, that there is no non-determinism as in the π-calculus. The choice-operator has been removed. Encryption protocols are deterministic, so there is no need for this power of expression.

2.1.3 Agents

Intuitively, before a process can send or receive anything on a channel, it has to get hold of the channel and *then* it is able to send or receive on it. That is, a process first *commits* to using a channel and then communicates. The process commits to a so-called *agent*.

An agent can be a process, an abstraction or a concretion. The syntax of agents is defined by the following grammar:

$$
\begin{aligned}
A, B &::= P \mid C \mid F \\
F, G &::= (x)P \\
C, D &::= (\nu\vec{m})\langle M\rangle P
\end{aligned}
$$

$(x)P$ is called an *abstraction.* This is the type of agent a process commits to before receiving a term on a channel; the term received is then bound to x. Sometimes, when convenient, we use abstractions to denote parameterized processes. This is the case in some of our examples. For example, if $S \triangleq (x)P$, we let $S(M) = P[M/x]$.

$(\nu\vec{m})\langle M\rangle P$ is called a *concretion.* A concretion is immediately able to output the term M. If the concretion does not generate any new names, i.e. $\{\vec{m}\} = \emptyset$, we write $(\nu)\langle M\rangle P$. Concretions are introduced to handle output in the same way as input, and to achieve a labelled transition system.

In what follows, Ag will denote the set of closed agents.

2.2 Formal semantics

The semantics of the spi calculus is similar to the one of the π-calculus mentioned in [Mil91]. In what follows, we present a labelled commitment semantics of the calculus. Another unlabelled semantics based on a reaction relation exists for the spi calculus, but the commitment relation characterizes it inductively and is better suited for proofs[AG98a]. Furthermore, the definition of framed bisimilarity is given in terms of the commitment relation.

2.2.1 The reduction relation

As a tool for the definition of the commitment relation we define the reduction relation. It can be regarded as a relation allowing processes to unfold and make preparations for a reaction.

$$\begin{aligned}
!P &> P \mid !P \\
[M \,\mathrm{is}\, M]P &> P \\
\mathrm{let}(x,y) = (M,N) \,\mathrm{in}\, P &> P[M/x][N/y] \\
\mathrm{case}\, 0 \,\mathrm{of}\;\; 0 : P \,\mathrm{suc}(x) : Q &> P \\
\mathrm{case}\, \mathrm{suc}(M) \,\mathrm{of}\;\; 0 : P \,\mathrm{suc}(x) : Q &> Q[M/x] \\
\mathrm{case}\, \{M\}_N \,\mathrm{of}\;\; \{x\}_N \,\mathrm{in}\, P &> P[M/x]
\end{aligned}$$

The definition of structural equivalence for the spi-calculus contains the reduction relation, but the commitment relation is, contrary to the usual presentation of the π-calculus' semantics, only based on the reduction relation[AG98a].

2.2.2 The commitment relation

First, the restriction and composition operators are extended to general agents.

$$\begin{aligned}
(\nu n)(x)P &\triangleq (x)(\nu n)P \\
Q \mid (x)P &\triangleq (x)(Q \mid P) \\
(\nu n)(\nu\vec{m})\langle M\rangle P &\triangleq \begin{cases} (\nu n,\vec{m})\langle M\rangle P & \text{if } n \in \mathrm{fn}[\![M]\!] \\ (\nu\vec{m})\langle M\rangle(\nu n)P & \text{otherwise} \end{cases} \\
Q \mid (\nu\vec{m})\langle M\rangle P &\triangleq (\nu\vec{m})\langle M\rangle(Q \mid P)
\end{aligned}$$

where we assume $x \notin \mathrm{fv}[\![Q]\!]$, $n \notin \{\vec{m}\}$ and $\{\vec{m}\} \cap \mathrm{fn}[\![Q]\!] = \emptyset$. The dual composition $A \mid Q$ is defined symmetrically.

For an abstraction $F = (x)P$ and concretion $C = (\nu\vec{n})\langle N\rangle Q$ we define the *interaction* between F and C as

$$\begin{aligned}
C@@F &\triangleq (\nu\vec{n})(Q \mid P[N/x]), \\
F@@C &\triangleq (\nu\vec{n})(P[N/x] \mid Q),
\end{aligned}$$

when $\{\vec{n}\} \cap \mathrm{fn}[\![P]\!] = \emptyset$. An interaction can be interpreted as a communication over a given channel between a sender and a receiver.

An *action* α is a name, a co-name or the unobservable action τ.

The *commitment transition system* $(Proc, \xrightarrow{\alpha}, Ag)$ has its transition relation defined inductively by

$$m(x).P \xrightarrow{m} (x)P \qquad \overline{m}\langle M\rangle.P \xrightarrow{\overline{m}} (\nu)\langle M\rangle P$$

$$\frac{P \xrightarrow{m} F \quad Q \xrightarrow{\overline{m}} C}{P \mid Q \xrightarrow{\tau} F@@C} \qquad \frac{P \xrightarrow{\overline{m}} C \quad Q \xrightarrow{m} F}{P \mid Q \xrightarrow{\tau} C@@F}$$

$$\frac{P \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} A \mid Q} \qquad \frac{Q \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} P \mid A}$$

$$\frac{P \xrightarrow{\alpha} A \quad \alpha \notin \{m, \overline{m}\}}{(\nu m)P \xrightarrow{\alpha} (\nu m)A} \qquad \frac{P > Q \xrightarrow{\alpha} A}{P \xrightarrow{\alpha} A}$$

Note that the last rule defines the commitment relation up to reduction on the left side and not up to structural equivalence as usual in the presentation of the semantics of the π-calculus.
The commitment relation for unobservable actions τ is a binary relation, $\xrightarrow{\tau} \subseteq Proc \times Proc$. We denote its reflexive and transitive closure by $\xrightarrow{\tau}^*$.

2.3 Testing equivalence

We want two processes to be testing equivalent, when they can resist the same attacks from the environment. That is, the two processes should in some way be indistinguishable to the attacker. This is essentially may-testing as in [NH83]. The notion of indistinguishability relies on the concept of a barb.

Definition 1 A *barb* is a name or a co-name.

Definition 2 Two closed processes P and Q are *testing equivalent*, written $P \simeq Q$, iff for all closed processes R and all barbs β:

$$P \mid R \xrightarrow{\tau}^* P' \xrightarrow{\beta} A \text{ for some } P' \text{ and } A \text{ iff}$$
$$Q \mid R \xrightarrow{\tau}^* Q' \xrightarrow{\beta} B \text{ for some } Q' \text{ and } B$$

Testing equivalence is sometimes also called attack-equivalence and can be used to check safety properties of protocols.

2.3.1 Expressing correctness properties

Generally, two properties are desired by the designer of a cryptographic protocol – secrecy and authenticity. Let P and Q be principals in a protocol.

- **(Secrecy)** If P sends an intended secret message M to Q, the message M is not revealed to the environment. Equivalently, one could say that the environment cannot distinguish whether the term M or another term M' was communicated between P and Q.
- **(Authenticity)** P and Q only respond to messages intended for each other. An attacker should not be able to make P or Q respond to some other message. Equivalently, one could say that the protocol should be equivalent to a specification that mimics all the communication in the protocol. In this specification the receiver magically only responds to the intended message.

A simple example, for which we formulate these properties, is the protocol that sends one encrypted message over a public channel:

$$\begin{aligned} A(M) &\triangleq \overline{c}\langle\{M\}_K\rangle.\mathbf{0} \\ B &\triangleq c(x).F(x) \\ Sys(M) &\triangleq (\nu K)(A(M) \mid B) \end{aligned}$$

The specification of the protocol is as follows:

$$\begin{aligned} A(M) &\triangleq \overline{c}\langle\{M\}_K\rangle.\mathbf{0} \\ B_{spec}(M) &\triangleq c(x).F(M) \\ S_{spec}(M) &\triangleq (\nu K)(A(M) \mid B_{spec}(M)) \end{aligned}$$

Proving secrecy would be the same as checking if $S(M) \simeq S(M')$ for all M, M', and proving authenticity would be the same as checking if $S(M) \simeq S_{spec}(M)$ for all M, M'.

3 Framed bisimilarity

This section defines a bisimulation equivalence introduced by Abadi and Gordon, called *framed bisimulation*[AG98b].

3.1 Fundamental definitions

The basic idea of framed bisimulation is the notion of an *environment*. We do not just relate two processes P and Q, but we relate them given an environment.

Definition 3 A *frame* f is a finite set of names. A *theory* t is a finite set of pairs of terms (M, N). An *environment* is a frame-theory pair, $e = (f, t)$.

Intuitively the elements of the frame are the names from P and Q that the environments knows. If $(M, N) \in t$ it means that the environment cannot distinguish the term M coming from P and the term N coming from Q.

Normally we use the letters e, f and t to denote environments, frames and theories. If the letters have the same marks or indices, e.g. e', f' and t', they are connected like $e' = (f', t')$.

Definition 4 Let e be an environment. Two terms M and N are indistinguishable under e, written $e \vdash M \leftrightarrow N$, if it can be derived by the following semantics:

$$\text{(Ind Zero)}\ \frac{}{e \vdash 0 \leftrightarrow 0} \qquad \text{(Ind Var)}\ \frac{}{e \vdash x \leftrightarrow x}$$

$$\text{(Ind Frame)}\ \frac{n \in f}{e \vdash n \leftrightarrow n} \qquad \text{(Ind Theory)}\ \frac{(M, N) \in t}{e \vdash M \leftrightarrow N}$$

$$\text{(Ind Pair)}\ \frac{e \vdash M \leftrightarrow M' \quad e \vdash N \leftrightarrow N'}{e \vdash (M, N) \leftrightarrow (M', N')}$$

$$\text{(Ind Suc)}\ \frac{e \vdash M \leftrightarrow M'}{e \vdash \mathrm{suc}(M) \leftrightarrow \mathrm{suc}(M')}$$

$$\text{(Ind Enc)}\ \frac{e \vdash M \leftrightarrow M' \quad e \vdash N \leftrightarrow N'}{e \vdash \{M\}_N \leftrightarrow \{M'\}_{N'}}$$

There can be inconsistencies in an environment. Assume that the theory contains the pairs (M,N). The environment has seen M from process P and N from process Q, i.e. the environment assumes that M and N are alike and adds them to the theory. Now P sends M, and Q sends L, where $N \neq L$. Now, if we add the pair (M,L) to the theory, it means that the environment thinks that M is the same as two different terms. This can obviously not be true. An environment containing such a pair is not ok.

Now consider the case where $(\{M_1\}_{M_2}, \{N_1\}_{N_2}) \in t$ and M_2 is revealed. Again, what the theory contains are elements that the environment thinks are alike. But an attacker would be able to decrypt one of the terms, but not the other. This means that they have different keys and thus are distinguishable. Again, such an environment is not ok.

Furthermore we do not want anything but ciphertexts in the theory. Two pairs of non-ciphertexts are immediately distinguishable, unless they are the same. These three conditions are collected in the following formal definition.

Definition 5 Environment e is ok, written $e \vdash \text{ok}$, if:

1. $\forall (M,N) \in t$ it must hold that M is closed, $\exists M_1, M_2 : M = \{M_1\}_{M_2}$ and $\nexists N_2 : e \vdash M_2 \leftrightarrow N_2$. The converse must also hold for N.
2. whenever $(M,N) \in t$ and $(M',N') \in t$, $M = M'$ iff $N = N'$.

During the execution of a protocol, the environment never shrinks. All information obtained will continue to be known, i.e. an environment can only be extended. Let us define precisely what it means for one environment to be an extension of another.

Definition 6 Let e and e' be environments. We say e' extends e, written $e \leq e'$ iff $\forall M, N : e \vdash M \leftrightarrow N \Rightarrow e' \vdash M \leftrightarrow N$.

Note that this relation is reflexive and transitive, but neither symmetric or antisymmetric; there may exist pairs of environments that are not related.

Two useful results about environments, proved by Abadi and Gordon in [AG98c] are:

Proposition 7 Suppose $e' \vdash \text{ok}$. Then $e \leq e'$ iff $f \subseteq f'$, and $\forall (M,N) \in t : e' \vdash M \leftrightarrow N$.

Lemma 8 Suppose $e \vdash \text{ok}$ and $e \vdash M \leftrightarrow N$. Then

1. If $e \vdash M \leftrightarrow N'$ then $N = N'$.
2. If $e \vdash M' \leftrightarrow N$ then $M = M'$.

A *framed process pair* is a quadruple (f, t, P, Q), where $P, Q \in \mathit{Proc}$. If $\mathcal{R}$ is a set of framed process pairs, we write $e \vdash P\mathcal{R}Q$ when $(f,t,P,Q) \in \mathcal{R}$. A *framed relation* is a set $\mathcal{R}$ of framed process pairs, such that $e \vdash \text{ok}$ whenever $e \vdash P\mathcal{R}Q$. This yields the necessary tools to define a framed simulation.

Definition 9 A *framed simulation* is a framed relation $\mathcal{S}$ such that, whenever $e \vdash P\mathcal{S}Q$, the following three conditions hold

1. If $P \xrightarrow{\tau} P'$ then there exists a process Q' such that $Q \xrightarrow{\tau} Q'$ and $e \vdash P'\mathcal{S}Q'$.
2. If $P \xrightarrow{c} (x)P'$ and $c \in f$ then there exists an abstraction $(x)Q'$ with $Q \xrightarrow{c} (x)Q'$ and, for all sets $\{\vec{n}\}$ disjoint from $\text{fn}[\![P]\!] \cup \text{fn}[\![Q]\!] \cup f \cup \text{fn}(t)$ and all closed terms M and N, if $(f \cup \{\vec{n}\}, t) \vdash M \leftrightarrow N$ then $(f \cup \{\vec{n}\}, t) \vdash P'[M/x]\mathcal{S}Q'[N/x]$.
3. If $P \xrightarrow{\overline{c}} (\nu\vec{m})\langle M\rangle P'$, $c \in f$ and $\{\vec{m}\} \cap (\text{fn}[\![P]\!] \cup \text{fn}(\pi_1(t)) \cup f) = \emptyset$ then there exists a concretion $(\nu\vec{n})\langle N\rangle Q'$ with $Q \xrightarrow{\overline{c}} (\nu\vec{n})\langle N\rangle Q'$ and $\{\vec{n}\} \cap (\text{fn}[\![Q]\!] \cup \text{fn}(\pi_2(t)) \cup f) = \emptyset$. Furthermore $\exists e' : e \leq e'$, $e' \vdash M \leftrightarrow N$, and $e' \vdash P'\mathcal{S}Q'$.

Notice that framed simulation is a late simulation as defined in [Mil91]. Finally,

Definition 10 A *framed bismulation* is a framed simulation $\mathcal{S}$ such that $\mathcal{S}^{-1} = \{e' \vdash Q\mathcal{S}P \mid e \vdash P\mathcal{S}Q \wedge e' = (f, \{(M,N) \mid (N,M) \in t\})\}$ is also a framed simulation.

Definition 11 Framed bisimilarity is the greatest framed bisimulation, written $\sim_f$.

Framed bisimulation implies testing equivalence. This is the major result of [AG98c] and our basis for studying framed bismilarity.

Theorem 12 Consider any closed processes P and Q, and any name $n \notin \mathrm{fn}[\![P]\!] \cup \mathrm{fn}[\![Q]\!]$. If $(\mathrm{fn}[\![P]\!] \cup \mathrm{fn}[\![Q]\!] \cup \{n\}, \emptyset) \vdash P \sim_f Q$ then $P \simeq Q$.

Unfortunately the converse implication does not hold. As an example of two processes that are testing equivalent, but not framed bisimilar, consider the following two:

$$P \triangleq (\nu c)(\mathbf{0} \mid \mathbf{0})$$
$$Q \triangleq (\nu c)(\overline{c}\langle 0\rangle.\mathbf{0} \mid c(x).\mathbf{0})$$

Trivially neither P nor Q exhibits any barbs, since c is restricted. Looking at the definition of test equivalence, we see that if P in parallel with a process exhibits a barb after a series of commitments and communications, so should Q in parallel with the same process. But since they do not exhibit any barbs, this holds. So they are test equivalent.

Looking at the definition of framed bisimilarity we see that if process P can make a τ transition, so should Q and vice versa. But Q can make a τ transition c and P cannot. Therefore they are not framed bisimilar.

As mentioned in the introduction, bisimulations are generally suitable for automatic verification, but unfortunately framed bisimulation is not one of them. The reason is, that the definition of framed bisimulation, or rather the parts concerning processes committing to either concretions or abstractions, contain quantifiers that range over infinite domains. Thus an algorithm cannot use brute force, when checking whether two processes are testing equivalent. It is the purpose of this article to eliminate one of these quantifiers, and replace it with an easily calculable function. Thus bringing us one step closer to automatic verification of the classic safety properties for cryptographic protocols.

4 Fenced bisimilarity

In this section we eliminate the need for the existential quantifier over environments, found in the conditions for concretions of framed bisimilarity. This is done by building a function that, given appropriate input, will return an environment that fulfills the given conditions. By careful design, this function will allow us to define an alternative form of framed bisimilarity, called fenced bisimilarity, which is more suited for automatic verification.

4.1 Fundamental definitions

Consider an environment e and two processes P and Q for which it holds that $P \xrightarrow{\overline{c}} (\nu\vec{m})\langle M\rangle P', c \in f, \{\vec{m}\} \cap (\mathrm{fn}[\![P]\!] \cup \mathrm{fn}(\pi_1(t)) \cup f) = \emptyset, Q \xrightarrow{\overline{c}} (\nu\vec{n})\langle N\rangle Q'$ and$\{\vec{n}\} \cap (\mathrm{fn}[\![Q]\!] \cup \mathrm{fn}(\pi_2(t)) \cup f) = \emptyset$. Looking at the definition of framed bisimilarity we see that in order for $e \vdash P \sim_f Q$, it must hold that $\exists e' : e \leq e' \vdash \mathrm{ok}$, $e' \vdash M \leftrightarrow N$ and $e' \vdash P' \sim_f Q'$.

In other words, in order for P and Q to be framed bisimilar under e, it must be possible to expand e in a way, such that the new environment e' is ok and cannot distinguish between the output M coming from P and the output N coming from Q. Furthermore under e' the remaining processes P' and Q' need to be framed bisimilar.

If we could somehow build e' with a function, depending on e, M and N, such that these requirements hold, then we could discard the existential quantifier and simplify the definition of framed bisimilarity.

In this section we define such a function, called the *expand* function ξ. Furthermore, we define the notion of fenced bisimilarity, which is variant of framed bisimilarity, using ξ. In later sections we shall prove that the notion of fenced and framed bisimilarity coincide.

Given that ξ exist, we can define the concept of fenced bisimilarity as follows.

Definition 13 A *fenced simulation* is a framed relation $\mathcal{T}$ such that, whenever $e \vdash P\mathcal{T}Q$, the following three conditions hold

1. If $P \xrightarrow{\tau} P'$ then there exists a process Q' such that $Q \xrightarrow{\tau} Q'$ and $e \vdash P'\mathcal{T}Q'$.
2. If $P \xrightarrow{c} A \equiv (x)P'$ and $c \in f$ then there exists an abstraction $B \equiv (x)Q'$ such that $Q \xrightarrow{c} B$ and, for all sets $\{\vec{n}\}$ disjoint from $\mathrm{fn}[\![P]\!] \cup \mathrm{fn}[\![Q]\!] \cup f \cup \mathrm{fn}(t)$ and all closed terms M and N, if $(f \cup \{\vec{n}\}, t) \vdash M \leftrightarrow N$ then $(f \cup \{\vec{n}\}, t) \vdash P'[M/x]\mathcal{T}Q'[N/x]$.
3. If $P \xrightarrow{\overline{c}} A \equiv (\nu\vec{m})\langle M\rangle P'$, $c \in f$ and $\{\vec{m}\} \cap (\mathrm{fn}[\![Q]\!] \cup \mathrm{fn}(\pi_1(t)) \cup f) = \emptyset$ then there is a concretion $B \equiv (\nu\vec{n})\langle N\rangle Q'$ such that $Q \xrightarrow{\overline{c}} B$, the set $\{\vec{n}\}$ is disjoint from $\mathrm{fn}[\![Q]\!] \cup \mathrm{fn}(\pi_2(t)) \cup f$ and $\xi(e, N, M) \vdash P'\mathcal{T}Q'$

Definition 14 A *fenced bismulation* is a fenced simulation $\mathcal{T}$ such that $\mathcal{T}^{-1}$ is also a fenced simulation.

Definition 15 *Fenced bisimilarity* $\sim_{\#}$ is the greatest fenced bisimulation.

But what should the expand function ξ look like? From a clearly theoretical point of view it would be enough to know that ξ exists. However, if the process of verifying that two processes are testing equivalent should ever be automated, we need a constructive approach.

The basic design idea of the ξ-function is to make a minimal extension of the environment by recursive calls, and its definition is found in Definition 16. The interesting cases of the algorithm occur when we add something to the environment. That is in the cases where $M = N = n$ and $M = \{M_1\}_{M_2}$ and $N = \{N_1\}_{N_2}$.

If $M = n = N$ it means that both the processes have sent out n. The only thing we can do is to add this n to the frame (Line 5). This is the only way to make the environment unable to distinguish a name. But this might violate the fact that our new environment is ok, so we need to repair the environment after putting n in the frame. This is done by moving all elements of the theory which violate the ok condition to a set of leftover pairs λ (Lines 7-10). Having done this, we are no longer assured that the terms which were indistinguishable in e remain indistinguishable in the new environment, so for each of the pairs in λ we call ξ to make these pairs indistinguishable again (Lines 11-12).

In the case that $M = \{M_1\}_{M_2}$ and $N = \{N_1\}_{N_2}$ there are two possibilities, depending on whether the keys are already indistinguishable or not. If they are, we simply call ξ on M_1 and N_1 (Line 14). That way N and M become indistinguishable. If not, we put the pair in the theory and repair the violations of the ok-condition as before (Line 16-25).

Because a pair is added to the theory, it is now also possible to violate condition 2 of definition 5, thus the lines 16-17 check this condition. By Lemma 8, a ciphertext cannot be indistinguishable from two different ciphertexts in an ok environment.

Without further ado, here comes ξ (Definition 16):

Definition 16

1 $\xi(e, M, N) \triangleq$
2 if $(e \vdash M \leftrightarrow N)$ then return(e)
3 case (N, M) of
4 $\quad [M = suc(L) \wedge N = suc(L')]\ \ e_\xi := \xi(e, L, L')$
5 $\quad [M = N = n]\ \ e_\xi := (f \cup \{n\}, t)$
6 $\quad\quad \lambda := \emptyset$
7 $\quad\quad$ forall $(\{M_1\}_{M_2}, \{N_1\}_{N_2}) \in t_\xi$ do
8 $\quad\quad\quad$ if $\exists L : (e_\xi \vdash M_2 \leftrightarrow L) \vee (e_\xi \vdash L \leftrightarrow N_2)$ then
9 $\quad\quad\quad\quad e_\xi := (f_\xi, t_\xi \backslash \{(M, N)\})$
10 $\quad\quad\quad\quad \lambda := \lambda \cup \{(M, N)\}$
11 $\quad\quad$ forall $(\{M_1\}_{M_2}, \{N_1\}_{N_2}) \in \lambda$ do
12 $\quad\quad\quad e_\xi := \xi(\xi(e_\xi, M_2, N_2), M_1, N_1)$
13 $\quad [M = \{M_1\}_{M_2} \wedge N = \{N_1\}_{N_2}]$
14 $\quad\quad$ if $(e \vdash M_2 \leftrightarrow N_2)$ then $e_\xi := \xi(e, M_1, N_1)$
15 $\quad\quad$ else
16 $\quad\quad\quad$ if $\exists (O, O') \in t : O = M \not\Leftrightarrow O' = N$ then
17 $\quad\quad\quad\quad$ return$(\perp)$
18 $\quad\quad\quad e_\xi := (f, t \cup \{(M, N)\})$
19 $\quad\quad\quad \lambda := \emptyset$
20 $\quad\quad\quad$ forall $(\{O_1\}_{O_2}, \{O'_1\}_{O'_2}) \in t_\xi$ do
21 $\quad\quad\quad\quad$ if $\exists L : (e_\xi \vdash O_2 \leftrightarrow L) \vee (e_\xi \vdash L \leftrightarrow O'_2)$ then
22 $\quad\quad\quad\quad\quad e_\xi := (f_\xi, t_\xi \backslash \{(O, O')\})$
23 $\quad\quad\quad\quad\quad \lambda := \lambda \cup \{(O, O')\}$
24 $\quad\quad\quad$ forall $(\{O_1\}_{O_2}, \{O'_1\}_{O'_2}) \in \lambda$ do
25 $\quad\quad\quad\quad e_\xi := \xi(\xi(e_\xi, O_2, O'_2), O_1, O'_1)$
26 $\quad [M = (M_1, M_2) \wedge\ N = (N_1, N_2)]$
27 $\quad\quad e_\xi := \xi(\xi(e, M_1, N_1), M_2, N_2)$
28 $\quad$ [otherwise] return$(\perp)$
29 return(e_ξ)

We define $\perp$ as the environment that is not ok and not related to any other environment. We say that ξ is *undefined* if it returns $\perp$.

The if-clause in line 2 catches the two trivial cases $M = N = 0$ and $M = N = x$. Note that the if-checks of the predicates in line 2, 8, 14, 16, and 21 can easily be implemented by algorithms. Seen from a programmer's perspective, the aesthetics of the algorithm could be improved by moving lines 16, 17, and 19-25 into a repair-function. Instead of lines 6-12 and 14-25, we would first add to either frame or theory in e_ξ and then set $e_\xi :=$ repair(e_ξ). Although this appears more aesthetically pleasing, the above description is more suited for proving properties of the ξ-function.

After presenting the ξ-function the question arises as to how the notion of fenced bisimilarity relates to that of framed bisimilarity.

4.2 Properties of fenced bisimilarity

Let us recapitulate the requirements of the expand function ξ. First of all, if ξ is well defined on a given input, then it should return a valid environment, as described in section 4. But ξ should also be complete, that is if there exist such a valid environment, then ξ should be we well defined.

In this section we prove that ξ really fulfills these demands. Actually we prove a somewhat stronger result, namely that not only does ξ return a valid extension whenever possible, it actually returns the smallest possible valid extension of the environment.

4.2.1 Soundness

To show soundness of fenced bisimilarity the following auxilliary lemma is necessary.

Lemma 17 Let $e \vdash \text{ok}$. If $\xi(e, M, N) \neq \bot$ then $e \leq \xi(e, M, N) \vdash \text{ok}$ and $\xi(e, M, N) \vdash M \leftrightarrow N$.

Proof The proof will be by induction in the number of extra ξ-calls used to obtain $\xi(e, M, N)$, written $\|\xi(e, M, N)\|$. Assume the lemma holds for all calculations $\xi(e_c, M_c, N_c)$, with $\|\xi(e_c, M_c, N_c)\| < i$. We must show that it holds for $\xi(e, M, N)$, when $\|\xi(e, M, N)\| = i$, therefore assume $\xi(e, M, N) \neq \bot$. If $e \vdash M \leftrightarrow N$ then $\xi(e, M, N) = e$, and trivially $e \leq \xi(e, M, N)$, $\xi(e, M, N) \vdash \text{ok}$ and $\xi(e, M, N) \vdash M \leftrightarrow N$. Assume therefore $e \vdash M \not\leftrightarrow N$ in the rest of the proof. Depending on the structure of M we have six cases to consider, albeit it reduces to four since $M = 0$ and $M = x$ are caught by the above.

- $[M = \text{suc}(L)]$ As $\xi(e, M, N) \neq \bot$ line 4 of the ξ algorithm yields $N = \text{suc}(L')$ and $\xi(e, M, N) = \xi(e, L, L')$. Now $\|\xi(e, L, L')\| < i$ which by use of the induction hypothesis and rule (Ind Suc) yields $e \leq \xi(e, M, N)$, $\xi(e, M, N) \vdash \text{ok}$ and $\xi(e, M, N) \vdash M \leftrightarrow N$.

- $[M = n]$ Since $\xi(e, M, N) \neq \bot$, $N = n$. Let us consider the individual steps of the algorithm in this case.

<u>After line 6.</u> $e_\xi := (f \cup \{n\}, t)$. Since $f \subseteq f_\xi$ and $t = t_\xi$, $e \leq e_\xi$ follows immediately by Proposition 7. Furthermore $n \in f_\xi$ yields $e_\xi \vdash n \leftrightarrow n$. However we do not necessarily have $e_\xi \vdash \text{ok}$ anymore. The following steps of the algorithm try to fix this.

<u>Before line 11.</u> We can now show $e_\xi \vdash \text{ok}$. Since $e \vdash \text{ok}$ and $t_\xi \subseteq t$ we know that all elements of t_ξ are ciphertexts and that we cannot have that $\exists(O_1, O_2), (O_1', O_2') \in t_\xi : O_1 = O_1' \not\Leftrightarrow O_2 = O_2'$. Moreover we removed from t_ξ all elements $(\{O_1\}_{O_2}, \{O_1'\}_{O_2'}) \in t_\xi$ for which $\exists L : e_\xi \vdash O_2 \leftrightarrow L \vee e_\xi \vdash L \leftrightarrow O_2'$, thus $e_\xi \vdash \text{ok}$.

At this point $e_\xi \vdash n \leftrightarrow n$ still holds, but not necessarily $e \leq e_\xi$. We know $\forall(M', N') \in t$ either $(M', N') \in t_\xi$ or $(M', N') \in \lambda$. In the latter case it is necessary to construct some kind of e_ξ' such that $e_\xi' \vdash M' \leftrightarrow N'$. Thus after line 12 is applied on (M', N') we must show $\xi_r = \xi(\xi(e_\xi, M_2', N_2'), M_1', N_1') \vdash M' \leftrightarrow N'$.

<u>Line 12.</u> Since we assumed $\xi(e, M, N) \neq \bot$ we have both $\xi(e_\xi, M_2', N_2') \neq \bot$ and $\xi_r \neq \bot$. Furthermore $\|\xi(e_\xi, M_2', N_2')\| < i$ and $\|\xi_r\| < i$ so the induction hypothesis yields $e_\xi \leq \xi(e_\xi, M_2', N_2') \vdash \text{ok}$, $\xi(e_\xi, M_2', N_2') \vdash M_2' \leftrightarrow N_2'$, $\xi(e_\xi, M_2', N_2') \leq \xi_r \vdash \text{ok}$ and $\xi_r \vdash M_1' \leftrightarrow N_1'$. Because $\xi_r \vdash M_2' \leftrightarrow N_2'$ and $\xi_r \vdash M_1' \leftrightarrow N_1'$ the rule (Ind Enc) gives $\xi_r \vdash M' \leftrightarrow N'$

<u>Line 29.</u> At this point $\xi(e, M, N) = e_\xi$. The construction of ξ forbids ξ to reduce the frame. Once all elements in λ have been through line 12 we have $f \subseteq f_\xi$ and $\forall(M', N') \in t : e_\xi \vdash M' \leftrightarrow N'$ thus by Proposition 7, $e \leq e_\xi$.

- $[M = \{M_1\}_{M_2}]$ Since $\xi(e, M, N) \neq \bot$, $N = \{N_1\}_{N_2}$. If $e \vdash M_2 \leftrightarrow N_2$ then $\xi(e, M, N) = \xi(e, M_1, N_1) \neq \bot$. Hence by the induction hypothesis $e \leq \xi(e, M, N) \vdash \text{ok}$ and also $\xi(e, M, N) \vdash M_1 \leftrightarrow N_1$, consequently $\xi(e, M, N) \vdash M \leftrightarrow N$.

If $e \vdash M_2 \not\leftrightarrow N_2$ we will consider the individual steps of the algorithm.

<u>After line 19.</u> $e_\xi = (f, t \cup \{(M, N)\})$. Since $f_\xi = f$ and $t \subseteq t_\xi$, $e \leq e_\xi$ follows immediately by Proposition 7. Furthermore $(M, N) \in t_\xi$ yields $e_\xi \vdash M \leftrightarrow N$. However, we are no longer sure that $e_\xi \vdash \text{ok}$. The following steps try to fix this.

<u>Before line 24.</u> We wish to prove that $e_\xi \vdash \text{ok}$. Since $e \vdash \text{ok}$, $t_\xi \subseteq t \cup \{(M, N)\}$, $M = \{M_1\}_{M_2}$ and $N = \{N_1\}_{N_2}$ we know all elements in t_ξ are ciphertexts. As $e \vdash \text{ok}$ and the only pair added to the theory was (M, N), the lines 16-23 could only have violated the second ok-condition if $\exists(O, O') \in t : O = M \not\Leftrightarrow O' = N$. In this situation $\xi(e, M, N) = \bot$ due to line 17, but this contradicts our $\xi(e, M, N) = \bot$ assumption. Moreover we have removed from t_ξ all elements $(\{O_1\}_{O_2}, \{O_1'\}_{O_2'}) \in t_\xi$ for which $\exists L : e_\xi \vdash O_2 \leftrightarrow L \vee e_\xi \vdash L \leftrightarrow O_2'$. Seeing that neither of the two ok-conditions are violated we conclude $e_\xi \vdash \text{ok}$.

Even though $e_\xi \vdash \text{ok}$ we do not necessarily have $e \leq e_\xi$ or $e_\xi \vdash M \leftrightarrow N$. For all $(M', N') \in t$ either $(M', N') \in t_\xi$ or $(M', N') \in \lambda$. In the latter case we would like to show that after applying line 25 on (M', N') then $\xi_r = \xi(\xi(e_\xi, M'_2, N'_2), M'_1, N'_1) \vdash M' \leftrightarrow N'$.

Line 25. Since we assumed $\xi(e, M, N) \neq \perp$ we have both $\xi(e_\xi, M'_2, N'_2) \neq \perp$ and $\xi_r \neq \perp$. Both $\|\xi(e_\xi, M'_2, N'_2)\| < i$ and $\|\xi_r\| < i$ therefore by the induction hypothesis, $e_\xi \leq \xi(e_\xi, M'_2, N'_2) \vdash \text{ok}$, $\xi(e_\xi, M'_2, N'_2) \vdash M'_2 \leftrightarrow N'_2$, $\xi(e_\xi, M'_2, N'_2) \leq \xi_r \vdash \text{ok}$ and $\xi_r \vdash M'_1 \leftrightarrow N'_1$. Because $\xi_r \vdash M'_2 \leftrightarrow N'_2$ and $\xi_r \vdash M'_1 \leftrightarrow N'_1$ the rule (Ind Enc) yields $\xi_r \vdash M' \leftrightarrow N'$.

Line 29. Now $\xi(e, M, N) = e_\xi$. By looking at the construction of ξ it is seen that it never can reduce the frame. Therefore $f \subseteq f_\xi$. Also $\forall (M', N') \in t : e_\xi \vdash M' \leftrightarrow N'$, hence $e \leq e_\xi$ by Proposition 7 after all elements in λ been through line 25. Likewise $e_\xi \vdash \text{ok}$.

• $[M = (M_1, M_2)]$ $\xi(e, M, N)$ can only be well-defined when $N = (N_1, N_2)$. In this case $\xi(e, M, N) = \xi(\xi(e, M_1, N_1), M_2, N_2)$, hence also $\xi(e, M_1, N_1) \neq \perp$. As $\|\xi(e, M_1, N_1)\| < i$ and $\|\xi(\xi(e, M_1, N_1), M_2, N_2)\| < i$ the induction hypothesis yields $e \leq \xi(e, M, N) \vdash \text{ok}$. Furthermore $\xi(e, M_1, N_1) \vdash M_1 \leftrightarrow N_1$, $\xi(e, M_1, N_1) \leq \xi(e, M, N) \vdash ok$ and $\xi(e, M, N) \vdash M_2 \leftrightarrow N_2$. Hence also $\xi(e, M, N) \vdash M \leftrightarrow N$ by rule (Ind Pair).

Thus we are done proving the lemma. □

Theorem 18 (Soundness) If $e \vdash P \sim_\# Q$ then $e \vdash P \sim_f Q$

Proof We show that $\sim_\#$ is a framed bisimulation. So consider an arbitrary framed process pair (e, P, Q) such that $e \vdash P \sim_\# Q$. It is enough to show that transitions of P can be matched by transitions of Q within $\sim_\#$.

Since the definition of $\sim_\#$ differs from $\sim_f$ only if $P \xrightarrow{\bar{c}} A \equiv (\nu \vec{m})\langle M \rangle P', c \in f$ and $\{\vec{m}\} \cap (\text{fn}[\![P]\!] \cup \text{fn}(\pi_1(t)) \cup f) = \emptyset$ we shall only consider this case. Since $e \vdash P \sim_\# Q$ we know that $\exists B \equiv (\nu \vec{n})\langle N \rangle Q' : Q \xrightarrow{\bar{c}} B$, $c \in f$, $\{\vec{n}\} \cap (\text{fn}[\![Q]\!] \cup \text{fn}(\pi_2(t)) \cup f) = \emptyset$ and $\xi(e, M, N) \vdash P' \sim_\# Q'$, provided that we can show that $e \leq \xi(e, M, N) \vdash \text{ok}$, $\xi(e, M, N) \vdash N \leftrightarrow M$ and $\xi(e, M, N) \vdash P' \sim_f Q'$. But these two conditions follow immediately from Lemma 17. □

4.2.2 Completeness

Two lemmas are necessary before showing the completeness result. The first lemma states that ξ yields the smallest valid extension.

Lemma 19 If $e \vdash \text{ok}$ and $\exists e' : e \leq e' \vdash \text{ok}$ and $e' \vdash M \leftrightarrow N$ then $\xi(e, M, N) \neq \perp$, $\xi(e, M, N) \vdash \text{ok}$ and $\xi(e, M, N) \leq e'$.

Proof Let $e \vdash \text{ok}$, $e \leq e' \vdash \text{ok}$ and $e' \vdash M \leftrightarrow N$. In order to show the lemma it suffices to prove that $\xi(e, M, N) \leq e'$, since this only holds if $\xi(e, M, N) \neq \perp$ which again by Lemma 17 implies $\xi(e, M, N) \vdash \text{ok}$.
The proof will use induction in the number of extra calls to ξ used to calculate $\xi(e, M, N)$, written $\|\xi(e, M, N)\|$. Assume the lemma holds for all calculations $\xi(e_c, M_c, N_c)$, with $\|\xi(e_c, M_c, N_c)\| < i$. We must show that it holds for $\xi(e, M, N)$, when $\|\xi(e, M, N)\| = i$. Depending on M we have six cases to consider

• $[M = 0]$ Since $e' \vdash \text{ok}$ and $e' \vdash M \leftrightarrow N$ we know $N = 0$ and also $e \vdash M \leftrightarrow N$. In this case $\xi(e, M, N) = e$, thus trivially $\xi(e, M, N) \leq e'$ because $e \leq e'$.

• $[M = x]$ Again, since $e' \vdash \text{ok}$ and $e' \vdash M \leftrightarrow N$, we have $N = x$. Furthermore we know $e \vdash M \leftrightarrow N$, thus looking at ξ we find that $\xi(e, M, N) = e \leq e'$.

• $[M = n]$ The fact that $e' \vdash \text{ok}$ and $e' \vdash M \leftrightarrow N$ yields $N = n$ and $n \in f'$. In this case $\xi(e, M, N)$ is calculated by an iterative algorithm. We therefore use invariants to prove our claim.

After line 6 We wish to show that $e_\xi \leq e'$. Since $e' \vdash \text{ok}$, Proposition 7 yields that $e_\xi \leq e'$ iff $f_\xi \subseteq f'$ and $\forall (M', N') \in t_\xi : e' \vdash M' \leftrightarrow N'$. Given that $f_\xi = f \cup \{n\}$, $n \in f'$ and $t_\xi = t$ it suffices to show that $f \subseteq f'$ and $\forall (M', N') \in t : e' \vdash M' \leftrightarrow N'$. However, this follows immediately by Proposition 7, since $e \leq e'$, thus $e_\xi \leq e'$.

Before line 11 Obviously $e_\xi \leq e'$, since we have only removed elements in t_ξ. However, we further wish to prove that $e_\xi \vdash \text{ok}$. Since $e \vdash \text{ok}$ and $t_\xi \subseteq t$ we know that all elements of t_ξ are ciphertexts and that we can't have that $\exists(O_1,O_2),(O_1',O_2') \in t_\xi : O_1 = O_1' \not\Leftrightarrow O_2 = O_2'$. Furthermore we have removed from t_ξ all elements $(\{O_1\}_{O_2},\{O_1'\}_{O_2'}) \in t_\xi$ for which $\exists L : e_\xi \vdash O_2 \leftrightarrow L \vee e_\xi \vdash L \leftrightarrow O_2'$, thus $e_\xi \vdash$ ok.

Now, let $O = \{O_1\}_{O_2}$, $O' = \{O_1'\}_{O_2'}$ and e_ξ^5 the e_ξ after line 5. If $(O,O') \in \lambda$, then $e_\xi^5 \vdash O \leftrightarrow O'$ and we know (wlog) that $\exists L : e_\xi^5 \vdash O_2 \leftrightarrow L$. Because $e_\xi^5 \leq e'$ this implies $e' \vdash O \leftrightarrow O'$ and $e' \vdash O_2 \leftrightarrow L$. Furthermore, since $e' \vdash$ ok, we have $(O,O') \notin t'$. Consequently $e' \vdash O_1 \leftrightarrow O_1'$ and $e' \vdash O_2 \leftrightarrow O_2'$.

Line 12 Because $e_\xi \vdash$ ok, $e_\xi \leq e' \vdash$ ok, $e' \vdash O_2 \leftrightarrow O_2'$ and $\|\xi(e_\xi,O_2,O_2')\| < i$ the induction hypothesis yields $\xi(e_\xi,O_2,O_2') \leq e'$. Likewise, since $\xi(e_\xi,O_2,O_2') \vdash$ ok, $\xi(e_\xi,O_2,O_2') \leq e' \vdash$ ok, $e' \vdash O_1 \leftrightarrow O_1'$ and $\|\xi(\xi(e_\xi,O_2,O_2'),O_1,O_1')\| < i$ the induction hypothesis yields $\xi(\xi(e_\xi,O_2,O_2'),O_1,O_1') \leq e'$. This holds for each application of line 12.

• $[M = \mathrm{suc}(L)]$ Given $e' \vdash$ ok and $e' \vdash M \leftrightarrow N$ we conclude $N = \mathrm{suc}(L')$ and $e' \vdash L \leftrightarrow L'$. In this case case $\xi(e,M,N) = \xi(e,L,L')$. But $\|\xi(e,L,L')\| < i$ so by induction we have $\xi(e,L,L') \leq e'$, thus $\xi(e,M,N) \leq e'$.

• $[M = (M_1,M_2)]$ Since $e' \vdash$ ok and $e' \vdash M \leftrightarrow N$ we know $N = (N_1,N_2)$, $e' \vdash M_1 \leftrightarrow N_1$ and $e' \vdash M_2 \leftrightarrow N_2$. In this case $\xi(e,M,N) = \xi(\xi(e,M_1,N_1),M_2,N_2)$. We know $\|\xi(e,M_1,N_1)\| < i$, hence by induction $\xi(e,M_1,N_1) \leq e'$ and thus $\xi(e,M_1,N_1) \vdash$ ok. Furthermore $\|\xi(\xi(e,M_1,N_1),M_2,N_2)\| < i$, so using the induction hypothesis a second time we find $\xi(\xi(e,M_1,N_1),M_2,N_2) \leq e'$.

• $[M = \{M_1\}_{M_2}]$ Using the facts that $e' \vdash$ ok and $e' \vdash M \leftrightarrow N$ we conclude $N = \{N_1\}_{N_2}$. Furthermore, either $e' \vdash M_1 \leftrightarrow N_1$ and $e' \vdash M_2 \leftrightarrow N_2$, or otherwise $(M,N) \in t'$.

If $e \vdash M_2 \leftrightarrow N_2$ then $\xi(e,M,N) = \xi(e,M_1,N_1)$. Furthermore, since $e \leq e'$, we have $e' \vdash M_2 \leftrightarrow N_2$, thus $(M,N) \notin t'$ because $e' \not\vdash$ ok otherwise. Therefore we must have $e' \vdash M_1 \leftrightarrow N_1$, but $\|\xi(e,M_1,N_1)\| < i$ so by induction $\xi(e,M_1,N_1) \leq e'$.

If $e \vdash M_2 \not\leftrightarrow N_2$ then $\xi(e,M,N)$ is calculated by an iterative algorithm. Again we use invariants to prove the lemma.

After line 19 Since $e' \vdash$ ok, Proposition 7 tells us that $e_\xi \leq e'$ iff $f_\xi \subseteq f'$ and $\forall(M',N') \in t_\xi : e' \vdash M' \leftrightarrow N'$. Given that $f_\xi = f$, $t_\xi = t \cup \{(M,N)\}$ and $e' \vdash M \leftrightarrow N$ it suffices to show that $f \subseteq f'$ and $\forall(M',N') \in t : e' \vdash M' \leftrightarrow N'$. However, this follows immediately by Proposition 7, since $e \leq e'$.

Before line 24 Obviously $e_\xi \leq e'$, because we only remove elements in t_ξ. We wish to prove that $e_\xi \vdash$ ok. Since $e \vdash$ ok, $t_\xi \subseteq t \cup \{(M,N)\}$, $M = \{M_1\}_{M_2}$ and $N = \{N_1\}_{N_2}$ we know all elements in t_ξ are ciphertexts. Furthermore we have removed from t_ξ all elements $(\{O_1\}_{O_2},\{O_1'\}_{O_2'}) \in t_\xi$ for which $\exists L : e_\xi \vdash O_2 \leftrightarrow L \vee e_\xi \vdash L \leftrightarrow O_2'$, thus only one thing can cause $e_\xi \not\vdash$ ok, namely if $\exists(O_1,O_2),(O_1',O_2') \in t_\xi : O_1 = O_1' \not\Leftrightarrow O_2 = O_2'$.

Now, assume to the contrary that $e_\xi \not\vdash$ ok. Because $e \vdash$ ok and $e_\xi \not\vdash$ ok we must have (wlog) that $\exists(M,X) \in t$, where $X \neq N$. The fact that $e \leq e'$ yields $e' \vdash M \leftrightarrow X$. But we also have $e' \vdash M \leftrightarrow N$, hence Lemma 8 yields $X = N$, which is a contradiction. Thus we have proved that $e_\xi \vdash$ ok.

Let $O = \{O_1\}_{O_2}$, $O' = \{O_1'\}_{O_2'}$ and e_ξ^{18} the e_ξ after line 18. If $(O,O') \in \lambda$, then $e_\xi^{18} \vdash O \leftrightarrow O'$ and we know (wlog) that $\exists L : e_\xi^{18} \vdash O_2 \leftrightarrow L$. Because $e_\xi^{18} \leq e'$ this implies $e' \vdash O \leftrightarrow O'$ and $e' \vdash O_2 \leftrightarrow L$. Furthermore, since $e' \vdash$ ok, we have $(O,O') \notin t'$. Consequently $e' \vdash O_1 \leftrightarrow O_1'$ and $e' \vdash O_2 \leftrightarrow O_2'$.

Line 27 Because $e_\xi \vdash$ ok, $e_\xi \leq e' \vdash$ ok, $e' \vdash O_2 \leftrightarrow O_2'$ and $\|\xi(e_\xi,O_2,O_2')\| < i$ the induction hypothesis yields $\xi(e_\xi,O_2,O_2') \leq e'$. Likewise, since $\xi(e_\xi,O_2,O_2') \vdash$ ok, $\xi(e_\xi,O_2,O_2') \leq e' \vdash$ ok, $e' \vdash O_1 \leftrightarrow O_1'$ and $\|\xi(\xi(e_\xi,O_2,O_2'),O_1,O_1')\| < i$ the induction hypothesis yields $\xi(\xi(e_\xi,O_2,O_2'),O_1,O_1') \leq e'$. This holds for every application of line 27.

Thus we are done proving the lemma. □

The second lemma states that when two processes are framed bisimilar under an extension of a given environment, then they are also fenced bisimilar under the least extension to this environment.

Lemma 20 If $e \vdash$ ok and $\exists e' : e \leq e' \vdash$ ok, $e' \vdash M \leftrightarrow N$ and $e' \vdash P \sim_f Q$ then $\xi(e,M,N) \vdash P \sim_\# Q$.

Proof We show that the framed relation $\mathcal{S} = \{(\xi(e,M,N),P,Q) \mid e \vdash \text{ok}, \exists e' : e \leq e', e' \vdash \text{ok}, e' \vdash M \leftrightarrow N, e' \vdash P \sim_f Q\}$ is a fenced bisimulation.

So consider an arbitrary framed process pair $(\xi(e,M,N),P,Q) \in \mathcal{S}$. We must show that transitions can match within $\mathcal{S}$.

We have that $e \vdash$ ok and that there exists an e' such that $e \leq e' \vdash$ ok, $e' \vdash M \leftrightarrow N$ and $e' \vdash P \sim_f Q$. That $\xi(e, M, N) \vdash$ ok follows immediately from Lemma 19.

We now consider the possible transitions of P and show that they can be matched by Q within $\mathcal{S}$. There are three cases to consider.

- $[P \xrightarrow{\tau} P']$ Since $e' \vdash P \sim_f Q$ we know that $\exists Q' : Q \xrightarrow{\tau} Q'$ and $e' \vdash P' \sim_f Q'$.

- $[P \xrightarrow{c} A \equiv (x)P', c \in f_\xi]$ By Lemma 19 we have that $f_\xi \subseteq f'$, so from $e' \vdash P \sim_f Q$ we conclude that $\exists B \equiv (x)Q' : Q \xrightarrow{c} B$ and $\forall \vec{n}, M', N' : \{\vec{n}\} \cap (\text{fn}[\![P]\!] \cup \text{fn}[\![Q]\!] \cup \text{fn}(t') \cup f') = \text{fv}[\![M']\!] = \text{fv}[\![N']\!] = \emptyset$ it holds that whenever $e'' \vdash M' \leftrightarrow N'$ then $e'' \vdash P'[M'/x] \sim_f Q'[N'/x]$, where $e'' = (f' \cup \{\vec{n}\}, t')$.

All we need to do now is to show that for such matches we have that $(\xi(e'', M', N'), P'[M'/x], Q'[N'/x]) \in \mathcal{S}$. It is enough to show that $e'' \vdash$ ok and that there is an extension e''' of e'' such that $e''' \vdash$ ok, $e''' \vdash M' \leftrightarrow N'$ and $e''' \vdash P'[M'/x] \sim_f Q'[N'/x]$. But this extension can simply be taken to be e'' itself.

- $[P \xrightarrow{\overline{c}} A \equiv (\nu\vec{m})\langle M'\rangle P'$ $, c \in f_\xi$ and $\{\vec{m}\} \cap(\text{fn}[\![P]\!] \cup \text{fn}(\pi_1(t')) \cup f') = \emptyset]$ As in the previous case, we have that $f_\xi \subseteq f'$. $e' \vdash P \sim_f Q$ now yields that $\exists B \equiv (\nu\vec{n})\langle N'\rangle Q' : Q \xrightarrow{\overline{c}} B$, $\{\vec{n}\} \cap (\text{fn}[\![Q]\!] \cup \text{fn}(\pi_2(t')) \cup f') = \emptyset$ and $\exists e'' : e' \leq e''$, $e'' \vdash M' \leftrightarrow N'$ and $e'' \vdash P' \sim_f Q'$.

Let $e_\xi = \xi(e, M, N)$. Since $e_\xi \leq e' \leq e''$ and $e_\xi \vdash$ ok, we can use e'' as the witness required for membership of $\mathcal{S}$ and get that $(\xi(e_\xi, M', N'), P', Q') \in \mathcal{S}$ as desired.

□

Theorem 21 (Completeness) If $e \vdash P \sim_f Q$ then $e \vdash P \sim_{\#} Q$.

Proof We show that $\sim_f$ is a fenced bisimulation. So consider an arbitrary framed process pair (e, P, Q). We must show that transitions of P can be matched by Q within $\sim_f$ according to the definition of fenced bisimulations.

The only interesting case occurs when $P \xrightarrow{\overline{c}} A \equiv (\nu\vec{m})\langle M\rangle P'$, $c \in f$ and $\{\vec{m}\} \cap (\text{fn}[\![P]\!] \cup \text{fn}(\pi_1(t)) \cup f) = \emptyset$. As $e \vdash P \sim_f Q$ we get $e \vdash ok$, $\exists B \equiv (\nu\vec{n})\langle N\rangle Q' : Q \xrightarrow{\overline{c}} B$, $\{\vec{n}\} \cap (\text{fn}[\![Q]\!] \cup \text{fn}(\pi_2(t)) \cup f) = \emptyset$ and $\exists e' : e \leq e' \vdash$ ok, $e' \vdash M \leftrightarrow N$ and $e' \vdash P' \sim_f Q'$.

We now only need to show $\xi(e, M, N) \vdash P' \sim_{\#} Q'$. But this follows immediately from Lemma 20. □

4.3 Examples

In order to show how to establish fenced bisimilarity results, we present some small example protocols focused on the secrecy property. The secrecy property is especially connected with output, where our notion of fenced bisimulation differs from that of framed bisimulation in [AG98b]. The ξ-function makes fenced bisimilarity more suited for automation; that is why we in the examples show *how* the fenced simulations are built, instead of just presenting one and proving its correctness.

Throughout this section M, M', M'', M_i, N_i denote closed terms; it is convenient to assume that no free names occur in them. I.e. they can be natural numbers, possibly as nested pairs. This restriction can be interpreted as modelling the actual binary data that a protocol tries to protect from an attacker.

None of the examples take the renaming of bound variables into account; this would only add unnecessary complexity. We simply assume that bound names do not clash with the free names, otherwise a simple α-conversion of the bound name is performed before the process is analyzed.

For the sake of convenience we introduce the syntactic abbreviation $\text{fn}(P, Q, t, f) \triangleq (\text{fn}[\![P]\!] \cup \text{fn}[\![Q]\!] \cup \text{fn}(t) \cup f)$.

4.3.1 Example 1

In this example we show that the two processes, $P \triangleq (\nu K)\,\overline{c}\langle\{M\}_K\rangle$ and $Q \triangleq (\nu K)\,\overline{c}\langle\{M'\}_K\rangle$ are fenced bisimilar under $e = (\{c, n\}, \emptyset)$. This is done by finding a fenced bisimulation $\mathcal{T}$ containing them. Hence $P \simeq Q$ follows by Theorem 12.

To begin with we construct $\mathcal{T}$ as a fenced simulation containing $e \vdash P\mathcal{T}Q$. Symmetric reasoning can be applied to show that $\mathcal{T}^{-1}$ is a fenced simulation as well. For now, let the only element in $\mathcal{T}$ be $e \vdash P\mathcal{T}Q$.

Since $t = \emptyset$ we immediately know that $e \vdash \text{ok}$. Now both P and Q can make a $\overline{c}$-commitment

$$P \xrightarrow{\overline{c}} (\nu K)\langle\{M\}_K\rangle\mathbf{0} \text{ and } Q \xrightarrow{\overline{c}} (\nu K)\langle\{M'\}_K\rangle\mathbf{0},$$

where $c \in f$ and $K \notin \text{fn}(P, Q, t, f)$. Therefore we only need to check if $\xi(e, \{M\}_K, \{M'\}_K) \vdash \mathbf{0}\mathcal{T}\mathbf{0}$. By following the steps of the ξ function we get $e_1 = \xi(e, \{M\}_K, \{M'\}_K) = (\{c, n\}, \{(\{M\}_K, \{M'\}_K)\})$. By adding $e_1 \vdash \mathbf{0}\mathcal{T}\mathbf{0}$, $\mathcal{T}$ will be a fenced simulation, since $e_1 \vdash \mathbf{0}\mathcal{T}\mathbf{0}$ is trivially fulfilled. Hence $e \vdash P \sim_{\#} Q$.

4.3.2 Example 2

This example illustrates the fact that the environment can compare ciphertexts without being able to decrypt them. Consider the two processes

$$P \triangleq (\nu K)\,\overline{c}\langle(\{M\}_K, \{M\}_K)\rangle$$
$$Q \triangleq (\nu K)\,\overline{c}\langle(\{M'\}_K, \{M''\}_K)\rangle$$

Let $e = (\text{fn}[\![P]\!] \cup \text{fn}[\![Q]\!] \cup \{n\}, \emptyset) = (\{c, n\}, \emptyset)$. Again we construct the fenced simulation $\mathcal{T}$ by starting out with $e \vdash P\mathcal{T}Q$.

By looking at $e \vdash P\mathcal{T}Q$ we see that both P and Q can make a $\overline{c}$-commitment

$$P \xrightarrow{\overline{c}} (\nu K)\langle(\{M\}_K, \{M\}_K)\rangle.\mathbf{0}$$
$$Q \xrightarrow{\overline{c}} (\nu K)\langle(\{M'\}_K, \{M''\}_K)\rangle.\mathbf{0}$$

and in both cases $K \notin \text{fn}(P, Q, t, f)$, thus we add $e_1 = \xi(e, (\{M\}_K, \{M\}_K), (\{M'\}_K, \{M''\}_K)) \vdash \mathbf{0}\mathcal{T}\mathbf{0}$. The ξ-call spawned by the first coordinate of the pairs returns $e_2 = (\{c, n\}, \{(\{M\}_K, \{M'\}_K)\})$. The ξ-call of the second coordinate will after line 18 have $e_\xi = (f_2, t_2 \cup \{(\{M\}_K, \{M''\}_K)\})$. If $M' = M''$, $e_1 = e_2$. If $M' \neq M''$ we will never be able to repair the environment, thus $e_1 = \perp$. In this case the two processes are not even testing equivalent. A separating process could be

$$R \triangleq c(z).\,\text{let}(x, y) = z\,\text{in}[x\,\text{is}\,y].\overline{d}\langle 1\rangle.$$

If R is put in parallel with P it will exhibit the barb $\overline{d}$ after the commitment. Put in parallel with Q it doesn't, because the two terms $\{M'\}_K$ and $\{M''\}_K$ are different. R shows how an attacker can distinguish between ciphertexts even without being able to decrypt them.

4.3.3 Example 3

In this example we look at a nested encryption where one of the keys is revealed. Let K_1 and K_2 be distinct and

$$P \triangleq (\nu K_1)(\nu K_2)\overline{c}\langle\{(M_1, \{M_2\}_{K_2})\}_{K_1}\rangle.\overline{c}\langle K_1\rangle$$
$$Q \triangleq (\nu K_1)(\nu K_2)\overline{c}\langle\{(N_1, \{N_2\}_{K_2})\}_{K_1}\rangle.\overline{c}\langle K_1\rangle$$

For $e = (\{c, n\}, \emptyset)$ we show $e \vdash P \sim_{\#} Q$. Let $\mathcal{T}$ be the fenced simulation we build by starting with $e \vdash P\mathcal{T}Q$.

Because $P \xrightarrow{\overline{c}} (\nu K_1, K_2)\langle\{(M_1, \{M_2\}_{K_2})\}_{K_1}\rangle\overline{c}\langle K_1\rangle$, $c \in f$, $\{K_1, K_2\} \notin \text{fn}(P, Q, t, f)$ and Q can make a similar commitment we must add and verify $e_1 \vdash \overline{c}\langle K_1\rangle\mathcal{T}\ \overline{c}\langle K_1\rangle$, where

$$e_1 = \xi(e, \{(M_1, \{M_2\}_{K_2})\}_{K_1}, \{(N_1, \{N_2\}_{K_2})\}_{K_1})$$

By following the steps of the ξ function we get $e_1 = (f, \{(\{(M_1, \{M_2\}_{K_2})\}_{K_1}, \{(N_1, \{N_2\}_{K_2})\}_{K_1})\})$.

The process $\overline{c}\langle K_1\rangle$ can perform another $\overline{c}$-commitment, $\overline{c}\langle K_1\rangle \xrightarrow{\overline{c}} \langle K_1\rangle\mathbf{0}$ which of course can be matched by Q. Thus we only need to add and verify $e_2 = \xi(e_1, K_1, K_1) \vdash \mathbf{0}\mathcal{T}\mathbf{0}$. ξ will in line 5 add K_1 to the frame, which compromises the keys of the only pair of the theory. The pair is removed and the repairing spawns another two ξ-calls in line 12. Since $(f \cup \{K_1\}, t) \vdash K_1 \leftrightarrow K_1$ the first call returns immediately and we are left with $\xi((f \cup \{K_1\}, t), (M_1, \{M_2\}_{K_2}), (N_1, \{N_2\}_{K_2}))$.

Because both M_1 and N_1 are plain terms, possibly nested as pairs, this call will only be well defined if $M_1 = N_1$. In this case $e_2 = (f \cup \{K_1\}, \{(\{M_2\}_{K_2}, \{N_2\}_{K_2})\})$. Trivially $e_2 \vdash \mathbf{0}\mathcal{T}\mathbf{0}$ and we are done. If $M_1 \neq N_1$ then $e \vdash P \not\sim_\# Q$. But neither $P \simeq Q$ since $R \triangleq c(x_1).c(x_2).\,\mathrm{case}\, x_1 \,\mathrm{of}\, \{w\}_{x_2} \,\mathrm{in}\; \mathrm{let}(y_1, y_2) = w \,\mathrm{in}\, [y_1 = M]\, \overline{d}\langle 1\rangle$ distinguishes them.

Notice how the theory, which was created by adding the two ciphertexts of the first commitment, could be repaired after the key K_1 was revealed. This might be considered a violation of the design criteria mentioned in [AG98b]. It states that one should never declare two terms as indistinguishable when later information allows the environment to distinguish between them. [AG98b] would instead prefer to add K_1 to the frame already in the first commitment, instead of adding $(\{(M_1, \{M_2\}_{K_2})\}_{K_1}, \{(N_1, \{N_2\}_{K_2})\}_{K_1})$ to the theory. The soundness and completeness results of our approach show that the $\leq$ notion is strong enough to cover our add-and-repair strategy. A strategy that does not require any far-sighted abilities. Such abilities are difficult to express in an algorithm.

4.3.4 Example 4

While the earlier examples were only concerned with output, this one is slightly more complicated since it actually models a true sender-receiver relationship. The protocol consists of two processes that share a key, and exchange a message. Formally the protocol is defined as follows:

$$\begin{aligned} A(M) &\triangleq \overline{c}\langle \{M\}_K\rangle.\mathbf{0} \\ B &\triangleq c(x).\,\mathrm{case}\, x \,\mathrm{of}\, \{w\}_K \,\mathrm{in}\, \mathbf{0} \\ S(M) &\triangleq (\nu K)(A(M) \mid B) \end{aligned}$$

To show that $S(M) \simeq S(M')$ it suffices to show that $e \vdash S(M) \sim_\# S(M')$, where $e = (\{c, n\}, \emptyset)$. Again we build a fenced simulation $\mathcal{T}$ starting with $e \vdash S(M)\mathcal{T}S(M')$.

$S(M)$ can make three different commitments.

- $[S(M) \xrightarrow{\tau} S_1(M)]$ $S_1(M) = (\nu K)(\mathbf{0} \mid \mathbf{0})$. Since $S(M')$ can make such a τ transition as well, we only have to check if $e \vdash S_1(M) \sim_\# \mathcal{S}_1(M')$. By simply adding it to $\mathcal{T}$ this holds, because $(\mathbf{0} \mid \mathbf{0})$ cannot perform any commitments.

- $[S(M) \xrightarrow{\overline{c}} (\nu K)\langle \{M\}_K\rangle S_2(M)]$ $S_2(M) = (\mathbf{0} \mid B)$. $c \in f$, $K \notin (\mathrm{fn}[\![S(M)]\!] \cup \mathrm{fn}(\pi_1(t)) \cup f)$. $S(M')$ can also make such a $\overline{c}$-transition, where $K \notin (\mathrm{fn}[\![S(M')]\!] \cup \mathrm{fn}(\pi_2(t)) \cup f)$. What is left to show is $e_1 = \xi(e, \{M\}_K, \{M'\}_K) \vdash (\mathbf{0} \mid B)\mathcal{T}(\mathbf{0} \mid B)$. Because $K \notin \{c, n\}$, $e_1 = (f, \{(\{M\}_K, \{M'\}_K)\}) \vdash \mathrm{ok}$ follows and $e_1 \vdash (\mathbf{0} \mid B)\mathcal{T}(\mathbf{0} \mid B)$ is added to $\mathcal{T}$.

  $(\mathbf{0} \mid B)$ can now make a c-commitment: $B \xrightarrow{c} (x)R$, where $c \in f_1$ and $R = (\mathbf{0} \mid \mathrm{case}\, x \,\mathrm{of}\, \{w\}_K \,\mathrm{in}\, \mathbf{0})$. To catch this, the set $(f_1 \cup \{\vec{n}\}, t_1) \vdash R[O/x]\mathcal{T}R[O'/x]$ is now added to $\mathcal{T}$ for all $O, O', \vec{n}$ which fulfill $\{\vec{n}\} \cap (\mathrm{fn}[\![B]\!]) \cup f_1 \cup \mathrm{fn}(t_1)) = \emptyset$, $\mathrm{fv}[\![O]\!] \cup \mathrm{fv}[\![O']\!] = \emptyset$ and $(f_1 \cup \{\vec{n}\}, t_1) \vdash O \leftrightarrow O'$. Since R cannot make any further commitments we are done with this case.

- $[S(M) \xrightarrow{c} (x)S_3(M)]$ $S_3(M) = (\nu K)(A(M) \mid \mathrm{case}\, x \,\mathrm{of}\, \{w\}_K \,\mathrm{in}\, \mathbf{0})$. Again $c \in f$. Let $e_1 = (f \cup \{\vec{n}\}, t)$, where $\{\vec{n}\} \cap \mathrm{fn}(S(M), S(M'), t, f) = \emptyset$. We have to ensure that if $e_1 \vdash O \leftrightarrow O'$ then $e_1 \vdash S_3(M)[O/x]\mathcal{T}S_3(M')[O'/x]$ for all O, O' such that and $\mathrm{fv}[\![O]\!] \cup \mathrm{fv}[\![O']\!] = \emptyset$.

  Therefore we add $e_1 \vdash S_3(M)[O/x]\mathcal{T}S_3(M')[O'/x]$ to $\mathcal{T}$ for all $O, O', \vec{n}$ which fulfill the above conditions.

  It is now necessary to investigate $S_3(M)[O/x]$. As it can only perform a $\overline{c}$ commitment: $S_3(M)[O/x] \xrightarrow{\overline{c}} (\nu K)\langle \{M\}_K\rangle R$, where $R = (\mathbf{0} \mid \mathrm{case}\, O \,\mathrm{of}\, \{w\}_K \,\mathrm{in}\, \mathbf{0})$, $c \in f_1$ and $K \notin (\mathrm{fn}[\![S_3(M)[O/x]]\!] \cup \mathrm{fn}(\pi_1(t_1)) \cup f_1$. The last condition is true, because $K \notin \{\vec{n}\}$ and $K \notin (\mathrm{fn}[\![O]\!] \cup \mathrm{fn}[\![O']\!])$ since otherwise $e_1 \vdash O \leftrightarrow O'$ could not hold. Of course $S_3(M)[O'/x]$ can do the same commitment, thus we only need to check if $e_3 = \xi(e_1, \{M\}_K, \{M'\}_K) \vdash R \leftrightarrow R$.

  Because R can't make any further commitment and $e_3 = (f_1, t_1 \cup \{(\{M\}_K, \{M'\}_K)\})$ it is sufficient to add $e_3 \vdash R\mathcal{T}R$.

The above construction demonstrates that the proof technique inherent in the definition of fenced bisimulation becomes involved when the processes contain input in the form of abstractions.

5 Conclusion and further work

In this paper we have defined an alternative characterization of the notion of framed bisimilarity and shown it to be sound and complete.

This alternative characterization of framed bisimilarity overcomes the minor of the two obstacles towards automatic proving of framed bisimilarity in the spi calculus. The quantifier in the output condition of framed bisimilarity was replaced by an algorithm. This enables us to construct the extended environment instead of just demanding its existence.

With our work it is possible to mechanize fenced bisimulation checking for processes which only perform τ or $\overline{c}$-commitments. To achieve secrecy and authenticity results for realistic protocols the mechanizing conjecture of [AG98b] remains to be proven for the input clause of fenced bisimilarity.

The obstacle here is the universal quantifier in the matching condition for abstractions. Abadi and Gordon's conjecture about the automation of framed bisimulation is built upon a conjecture that the input of unbound size and extension of the environment by unlimited amount of new names can be divided into behavioral equivalence classes[AG98b].

This is the underlying idea of the symbolic operational semantics for the π-calculus suggested by Boreale and De Nicola [BN94] and Lin [Lin94] and employed by Dam in [Dam97], so the search for a decision algorithm can probably take advantage of these results.

One should note, though, that one can only hope to find decision algorithms for fragments of the spi calculus because of its universal computing power. A good candidate for such a fragment is the finite-control fragment whose π-calculus counterpart has already been investigated by Lin [Lin94] and Dam [Dam97]. It remains to be seen which encryption protocols are expressible within this subcalculus.

However, there is yet another obstacle to be dealt with, if one wants to automate the verification of security protocols, namely the universal quantification over terms inherent in the definitions of secrecy and authenticity in Section 2.3.1. We conjecture that an application of the symbolic techniques such as to partition terms into finite many equivalence classes may prove useful here.

6 Acknowledgements

Discussions with Anders Rhod Gregersen, Thomas Illum Rasmussen, Thomas Ording and the members of the ν-club at the Department of Computer Science helped refine the ideas in this paper. A copy of the unpublished [AG98c] was provided to us by Martín Abadi.

References

[AG97a] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*. ACM Press, 1997.

[AG97b] Martín Abadi and Andrew D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In Antoni Mazurkiewicz and Józef Winkowski, editors, *CONCUR '97: Concurrency Theory, 8th International Conference*, volume 1243 of *Lecture Notes in Computer Science*, pages 59–73, Warsaw, Poland, 1–4 July 1997. Springer-Verlag.

[AG98a] M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. SRC Research Report 149, Digital Equipment Corp., 25 January 1998.

[AG98b] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. *Lecture Notes in Computer Science*, 1381:12–26, 1998.

[AG98c] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. Extended version of [AG98b] containing all proofs., January 27, 1998.

[AN95] R. Anderson and R. Needham. Programming Satan's computer. *Lecture Notes in Computer Science*, 1000:426–440, 1995.

[BAN89] Michael Burrow, Martín Abadi, and Roger Needham. A logic for authentication. Technical Report SRC-89-39, Syntax Research Center, Cowell College, University of California at Santa Cruz, February 28, 1989.

[BN94] Michele Boreale and Rocco De Nicola. A symbolic semantics for the π-calculus (extended abstract). In Bengt Jonsson and Joachim Parrow, editors, *CONCUR '94: Concurrency Theory, 5th International Conference*, volume 836 of *Lecture Notes in Computer Science*, pages 299–314, Uppsala, Sweden, 22–25August 1994. Springer-Verlag.

[Dam97] Mads Dam. On the decidability of process equivalences for the π-calculus. *Theoretical Computer Science*, 183:215–228, 1997. Previously published as SICS Research Report R94:20, 1994.

[Lin94] Huimin Lin. Symbolic bisimulation and proof systems for the π-calculus. Technical Report 7/94, School of Cognitive and Computing Sciences, University of Sussex, UK, 1994.

[Mil89] Robin Milner. *Communication and Concurrency.* Prentice Hall Europe, 1989.

[Mil91] Robin Milner. The polyadic π-calculus: a tutorial. Technical Report ECS–LFCS–91–180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK, October 1991. Appeared in *Proceedings of the International Summer School on Logic and Algebra of Specification*, Marktoberdorf, August 1991. Reprinted in *Logic and Algebra of Specification*, ed. F. L. Bauer, W. Brauer, and H. Schwichtenberg, Springer-Verlag, 1993.

[MMS97] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murφ. In *IEEE Symposium on Security and Privacy*, pages 141–151, Oakland, CA, 1997. IEEE Computer Society Press.

[MPW93] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149–171, June 1993.

[NH83] Rocco De Nicola and Matthew C. B. Hennessy. Testing equivalence for processes. In Josep Díaz, editor, *Automata, Languages and Programming, 10th Colloquium*, volume 154 of *Lecture Notes in Computer Science*, pages 548–560, Barcelona, Spain, 18–22 July 1983. Springer-Verlag.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of 5th GI Conference LNCS 104*, pages 167–183. Springer-Verlag, 1981.

[Sch96] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, Inc., 1996.