Deciding Hedged Bisimilarity

Alessio Mansutti[1,2⋆] and Marino Miculan[1⋆⋆]

[1]DMIF, University of Udine, Italy
[2]LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay, France

Abstract. The spi-calculus is a formal model which allows to reduce the verification of security properties of cryptographic protocols to the verification of observational equivalences between spi processes. In this paper we prove that *hedged bisimilarity*, which is equivalent to barbed equivalence, is decidable on finite spi-calculus processes. The algorithm we provide works with any term equivalence satisfying a simple set of conditions, thus encompassing many different encryption schemata.

1 Introduction

The spi-calculus is a formal model for the description and analysis of cryptographic protocols [1]. In this model, the verification of security properties can be reduced to the verification of suitable behavioural equivalences between spi processes; for instance, if $P(M)$ represents a system of processes exchanging message M, the system guarantees strong confidentiality of the message if an attacker cannot observe any effective difference between $P(M)$ and $P(M')$, for any M, M'.

To this end, several behavioural equivalences for the spi-calculus have been proposed; we refer to [3] for a detailed comparison. In particular, in *loc. cit.* Borgström and Nestmann defined *hedged bisimilarity*; this bisimilarity is shown to be equivalent to *barbed equivalence* [7] and hence it is well suited to capture the interaction of an attacker with the system and model his knowledge.

In this paper we prove that hedged bisimilarity (and hence barbed equivalence) is decidable on *finite* processes (i.e. processes without replication). Our result extends a similar one by Hüttel for framed bisimilarity, an equivalence strictly finer than hedged bisimilarity [5]. Moreover, our algorithm can be readily applied to different encryption/decryption schemata just by changing the congruence over terms, as long as some mild conditions are satisfied; these conditions are introduced in Section 2, where we recall also the syntax and semantics of spi-calculus. Another difference with respect to previous work is that in our formulation terms are typed and processes can inspect these types by means of type-checking predicates; this corresponds to the fact that observers (even attackers) can deduce the structure of a message, even if encrypted, from its size or from the encoding algorithm. Moreover, we adopt a *late* operational semantics, which allows for a simpler definition of hedged bisimilarity, as we will see in Section 3. In Section 4 we show that hedged bisimilarity is decidable on

⋆ Project completed at University of Udine before being contracted by CNRS.
⋆⋆ Partially supported by UniUd PRID 2017 *ENCASE*.

finite spi-calculus processes; to this end, we will introduce the notion of *canonical* processes as canonical representatives of congruence classes. Finite canonical processes can be analyzed to determine the (finite) space of terms and names which have to be considered at each step of the bisimulation game. Based on this result, we provide the algorithm for the verification of hedged bisimulation.

2 The spi-calculus

The *spi-calculus* extends the π-calculus terms with primitives for encryption and decryption. Formally, let $\mathcal{N}$ be a countable set of names (typically of channels) ranged over by $a, b, c, \ldots$, and $\mathcal{V}$ a countable set of variable symbols, ranged over by $x, y, z, \ldots$. The set of spi-calculus terms is given by the grammar

$$A ::= a \mid x \qquad t ::= A \mid (t_1, t_2) \mid \pi_1(t) \mid \pi_2(t) \mid \{t_1\}_{t_2} \mid \mathtt{D}_{t_2}(t_1)$$
$$\phi ::= \mathbf{true} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid [t_1 = t_2] \mid \mathit{isname}(t) \mid \mathit{ispair}(t) \mid \mathit{isenc}(t)$$

Intuitively, $\{t_1\}_{t_2}$ denotes the term t_1 encrypted using key t_2, and (t_1, t_2) is the pair composed by t_1 and t_2. The decryption $\mathtt{D}_{t_2}(t_1)$ opens t_1 using the key t_2, and $\pi_1(t)$, $\pi_2(t)$ are the two projections. We call the encryption and pairing operators *constructors*, and the projection and decryption operators *destructors*.

We denote by $\mathrm{fv}(t)$, $n(t)$ the sets of (free) variables and names in term t, respectively. As usual, a term t is *ground* if $\mathrm{fv}(t) = \emptyset$. A ground term is moreover said to be a *message* whenever no destructor appears in it. We denote with $\mathcal{T}$ and $\mathcal{G}$ respectively the sets of all terms and ground terms. $\mathcal{M}$ denotes the set of all messages, ranged over by $M, N, \ldots$. Lastly, ϕ ranges over boolean expression, where $[t_1 = t_2]$ is the equality test between terms and $\mathit{isname}(t)$, $\mathit{ispair}(t)$ and $\mathit{isenc}(t)$ check whether a term t is a name, a pair of an encryption respectively. These last three operators, not present in the original definition [1], reflect the fact that often a process (e.g., an attacker) can easily infer if a term is a name, a pair or an encrypted value, by observing its structure or by knowing how data are encoded (in accordance to Kerckhoffs's principle).

Moreover, we extend further the calculus by abstracting from the syntactical equality normally used for $[t_1 = t_2]$, see [1,3,5]. To do so we introduce a typing system for ground terms. Types are defined by the grammar:

$$\tau ::= \mathsf{N} \mid \mathsf{B} \mid \tau_1 \times \tau_2 \mid \mathsf{C}(\tau)$$

where N, B are the base types of names and boolean values respectively, $\tau_1 \times \tau_2$ is the pair type and $\mathsf{C}(\tau)$ is the type of encrypted terms of type τ. Figure 1 shows the rules for the typing judgment $t : \tau$ over ground terms. For well-typed terms t, we denote with $\mathrm{keys}(t)$ the set of names that are used as keys. Formally,

$$\mathrm{keys}(t) = \{\mathrm{key}(t_2) \mid \text{ there is } t_1 \text{ s.t. } \mathtt{D}_{t_2}(t_1) \text{ or } \{t_1\}_{t_2} \text{ are a subterm of } t\}$$

where $\mathrm{key}(a) = a$, $\mathrm{key}(\pi_i(t_1, t_2)) = \mathrm{key}(t_i)$ and $\mathrm{key}(\mathtt{D}_{t_3}(\{t_1\}_{t_2})) = \mathrm{key}(t_1)$.

Ground terms are taken up-to some computable structural congruence $\cong$. We can think of two terms in the same equivalence class of $\cong$ as being computationally equivalent, i.e. executing the encryption/decryption algorithms on them leads to the same value. With this novelty we aim to account for different types

$$\frac{a \in \mathcal{N}}{a : \mathsf{N}} \qquad \frac{}{\mathbf{true} : \mathsf{B}} \qquad \frac{\phi : \mathsf{B}}{\neg\phi : \mathsf{B}} \qquad \frac{\phi_1 : \mathsf{B} \quad \phi_2 : \mathsf{B}}{\phi_1 \wedge \phi_2 : \mathsf{B}} \qquad \frac{}{[t_1 = t_2] : \mathsf{B}}$$

$$\frac{t_1 : \tau_1 \quad t_2 : \tau_2}{(t_1, t_2) : \tau_1 \times \tau_2} \qquad \frac{t : \tau_1 \times \tau_2}{\pi_i(t) : \tau_i} i = 1, 2 \qquad \frac{t_1 : \tau \quad t_2 : \mathsf{N}}{\{t_1\}_{t_2} : \mathsf{C}(\tau)} \qquad \frac{t_1 : \mathsf{C}(\tau) \quad t_2 : \mathsf{N}}{\mathtt{D}_{t_2}(t_1) : \tau}$$

$$\frac{}{\mathit{isname}(t) : \mathsf{B}} \qquad \frac{}{\mathit{ispair}(t) : \mathsf{B}} \qquad \frac{}{\mathit{isenc}(t) : \mathsf{B}}$$

Fig. 1. Typing judgment of terms.

of encryption schemata, which can be expressed by considering different $\cong$ as long as, for any $t : \tau_1$ and $t' : \tau_2$ such that $t \cong t'$, it respects the three properties:

1. **Type equivalence** t and t' have the same type, i.e. $\tau_1 = \tau_2$;
2. **Key equivalence** the same names are used as keys and, by changing any of them with a fresh name we obtain two equivalent terms. Formally,
$$\mathrm{keys}(t) = \mathrm{keys}(t') \wedge \forall a \in \mathrm{keys}(t) \forall b \in \mathcal{N} \setminus (n((t, t'))) : \ t\{b/a\} \cong t'\{b/a\}$$
where $t\{b/a\}$ is the substitution replacing all occurrences of a in t with b.
3. **Deterministic decryption** erasing all encryptions and decryption and applying the same projections leads to the same term, i.e. after applying the rewriting rules $\pi_i((t_1, t_2)) \to t_i$, $\mathtt{D}_{t_2}(t_1) \to t_1$ and $\{t_1\}_{t_2} \to t_1$ in all subterms of t and t' we obtain two syntactically equivalent terms.

It is easy to check that the equivalence used in [1,3,5] respects all these conditions. Other encryption algorithms (and other abstract data types) can be considered by adapting the congruence relation, as long as it respects the properties above. For example commutative ciphers (like Elgamal encryption) can be considered by adding the axiom $\forall M \in \mathcal{G} \ \forall a, b \in \mathcal{N} : \ \{\{M\}_a\}_b \cong \{\{M\}_b\}_a$.

Definition 1 (Processes). *The spi-calculus processes are defined as follows:*

$$P ::= \ 0 \mid A(x).P \mid \overline{A}\langle t \rangle.P \mid P_1 | P_2 \mid P_1 + P_2 \mid (\nu \boldsymbol{m})P \mid !P \mid \phi.P \mid \mathtt{let}\ x = t\ \mathtt{in}\ P$$

where t and ϕ are respectively a term and a boolean formula, x is a variable, $\boldsymbol{m}$ is a list of names and A can be both a name or a variable.

We extend fv on processes, so that $\mathrm{fv}(P)$ is the set of free variables occurring in P, and denote with $\mathrm{fn}(P)$ the set of free names of P (i.e. names not occurring in the scope of a $(\nu \boldsymbol{m})$ operator). The syntax above is the usual one from π-calculus [8], with these differences:

- input/output operations exchange terms, not only names;
- $\phi.P$ is the *guard operator*, that behaves as P if the boolean formula ϕ holds;
- $\mathtt{let}\ x = t\ \mathtt{in}\ P$ is the *let operator* that computes the value of t, binds the variable x to it and then executes P.

Processes of the spi-calculus are taken up-to the usual structural congruence familiar from the π-calculus theory and shown in Figure 2.

$$(\nu \boldsymbol{m})0 \equiv 0 \qquad P|0 \equiv P \qquad !P \equiv P|!P \qquad P|Q \equiv Q|P \qquad (P|Q)|R \equiv P(Q|R)$$

$$P+Q \equiv Q+P \qquad (P+Q)+R \equiv P+(Q+R) \qquad (\nu \boldsymbol{m})(P|Q) \equiv (\nu \boldsymbol{m})P|Q \text{ if } m \notin \mathrm{fn}(Q)$$

$$(\nu \boldsymbol{m})(\nu \boldsymbol{n})P \equiv (\nu \boldsymbol{n})(\nu \boldsymbol{m})P \qquad \frac{P \text{ and } Q \text{ are } \alpha\text{-equivalent}}{P \equiv Q} \qquad \frac{P \equiv Q}{P|R \equiv Q|R}$$

$$\frac{P \equiv Q}{P+R \equiv Q+R} \qquad \frac{P \equiv Q}{(\nu \boldsymbol{m})P \equiv (\nu \boldsymbol{m})Q} \qquad \frac{P \equiv Q}{Q \equiv P} \qquad \frac{P \equiv Q \quad Q \equiv R}{P \equiv R}$$

Fig. 2. Structural congruence of processes

To define the operational semantics of the spi-calculus we need to evaluate ground terms and boolean expressions. The evaluation is defined over well-typed terms, where each type denotes a set of values. The interpretation of base types is defined as $\mathcal{I}(\mathsf{N}) = \mathcal{N}$ and $\mathcal{I}(\mathsf{B}) = \{\mathbf{true}, \mathbf{false}\}$, whereas for pairs and encryptions it is defined as:

$$\mathcal{I}(\mathsf{C}(\tau)) = \{\{M\}_a \mid M \in \mathcal{I}(\mathsf{C}(\tau)), a \in \mathcal{N}\}/\cong$$
$$\mathcal{I}(\tau_1 \times \tau_2) = \{(M_1, M_2) \mid M_1 \in \mathcal{I}(\tau_1), M_2 \in \mathcal{I}(\tau_2)\}/\cong$$

Definition 2 (Evaluation). *The* evaluation *for well-typed ground terms and boolean expressions is a partial function* $[\![\cdot]\!]$ *defined recursively as follows:*

$$[\![a]\!] = a \in \mathcal{N} \qquad [\![(t_1, t_2)]\!] = ([\![t_1]\!], [\![t_2]\!])$$
$$[\![\mathbf{true}]\!] = \mathbf{true} \qquad [\![\pi_1(t)]\!] = v_1 \text{ if } [\![t]\!] = (v_1, v_2)$$
$$[\![\phi_1 \wedge \phi_2]\!] = [\![\phi_1]\!] \wedge [\![\phi_2]\!] \qquad [\![\pi_2(t)]\!] = v_2 \text{ if } [\![t]\!] = (v_1, v_2)$$
$$[\![\neg\phi]\!] = \neg[\![\phi]\!] \qquad [\![\{t_1\}_{t_2}]\!] = \{[\![t_1]\!]\}_{[\![t_2]\!]}$$
$$[\![\mathtt{D}_{t_2}(t_1)]\!] = t \in \tau \text{ if } [\![t_2]\!] = a \in \mathcal{N} \text{ and } [\![t_1]\!] \cong \{t\}_a \in \mathcal{I}(\mathsf{C}(\tau))$$
$$[\![[t_1 = t_2]]\!] = \mathbf{true} \text{ if } [\![t_1]\!] \cong [\![t_2]\!];\ \mathbf{false} \text{ otherwise}$$
$$[\![\mathit{isname}(t_1)]\!] = \mathbf{true} \text{ if } [\![t_1]\!] \in \mathcal{N};\ \mathbf{false} \text{ otherwise}$$
$$[\![\mathit{ispair}(t_1)]\!] = \mathbf{true} \text{ if } [\![t_1]\!] \in \mathcal{I}(\tau_1 \times \tau_2) \text{ for some } \tau_1 \text{ and } \tau_2;\ \mathbf{false} \text{ otherwise}$$
$$[\![\mathit{isenc}(t_1)]\!] = \mathbf{true} \text{ if } [\![t_1]\!] \in \mathcal{I}(\mathsf{C}(\tau)) \text{ for some } \tau;\ \mathbf{false} \text{ otherwise}$$

Following [1,5] we define a *late input* operational semantics for spi-calculus processes. We first define the *reduction* relation, which describes how processes unfold and execute internal computations.

$$\frac{\phi \text{ well-typed} \quad [\![\phi]\!] = \mathbf{true}}{\phi.P > P} \qquad \frac{t \text{ well-typed} \quad [\![t]\!] \cong v}{\mathtt{let}\ x = t\ \mathtt{in}\ P > P\{v/x\}} \qquad \frac{P > P'}{P \equiv P'}$$

Then, the late operational semantics of the spi-calculus is represented by a labelled transition relation $P \xrightarrow{\alpha} Q$, where $\alpha \in \{a, \overline{a} \mid a \in \mathcal{N}\} \cup \{\tau\}$ is the name of a channel (for both input and output) or the silent transition τ (also called τ-transition), and Q ranges over processes, *concretions* and *abstractions* (see below). The rules of this operational semantics are in Figure 3.

$$\text{(input)}\frac{a \in \mathcal{N}}{a(x)P \xrightarrow{a} (x)P} \qquad \text{(output)}\frac{t \text{ well-typed} \quad [\![t]\!] \cong M \quad a \in \mathcal{N}}{\overline{a}\langle t\rangle P \xrightarrow{\overline{a}} (\nu)\langle M\rangle P}$$

$$\text{(interaction)}\frac{P \xrightarrow{n} (x)P' \quad Q \xrightarrow{\overline{n}} (\nu \boldsymbol{m})\langle M\rangle Q'}{P|Q \xrightarrow{\tau} (\nu \boldsymbol{m})(P'\{M/x\}|Q')}\{\boldsymbol{m}\} \cap \text{fn}(P') = \emptyset$$

$$\text{(restriction)}\frac{P \xrightarrow{\alpha} D \quad \alpha \notin \boldsymbol{m} \cup \overline{\boldsymbol{m}}}{(\nu m)P \xrightarrow{\alpha} (\nu m)D} \qquad \text{(parallel)}\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\text{(sum)}\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \text{(equivalence)}\frac{P \equiv Q \quad Q \xrightarrow{\alpha} Q' \quad Q' \equiv P'}{P \xrightarrow{\alpha} P'}$$

Fig. 3. Late operational semantics of the spi-calculus.

Concretions are "message-continuation" pairs resulting from an output: a process executing an output transition yields the concretion $(\nu)\langle M\rangle Q$ (where (ν) is a restriction on a empty set of names) which can be seen as a process ready to send a message and to continue as Q. On the other hand, an abstraction $(x)P$ is a "process with a hole", resulting from an input transition; it can be seen as process waiting to receive a message M and continue as $P\{M/x\}$. An abstraction $(x)P$ and a concretion $(\nu \boldsymbol{m})\langle M\rangle Q$, where $\boldsymbol{m}$ is a list of names, can synchronize resulting in a process where the message M is received by $(x)P$. In order to define this interaction, we need to extend restriction and parallel composition operators to abstractions and concretions, as follows:

$$(\nu \boldsymbol{m})(x)P \triangleq (x)(\nu \boldsymbol{m})P$$

$$(\nu a)(\nu \boldsymbol{m})\langle M\rangle P \triangleq \begin{cases} (\nu a, \boldsymbol{m})\langle M\rangle P & \text{if } n \in \text{fn}(M) \\ (\nu \boldsymbol{m})\langle M\rangle (\nu a)P & \text{otherwise} \end{cases}$$

$$R|(x)P \triangleq (x)(R|P) \quad \text{where } x \notin \text{fv}(R)$$

$$R|(\nu \boldsymbol{m})\langle M\rangle P \triangleq (\nu \boldsymbol{m})\langle M\rangle (R|P) \quad \text{where } \{\boldsymbol{m}\} \cap \text{fn}(R) = \emptyset$$

As usual, we denote with $P \Longrightarrow Q$ the transitive closure of the τ-transition whereas $P \overset{\alpha}{\Longrightarrow} Q$ denotes $P \Longrightarrow \xrightarrow{\alpha} Q$.

3 Hedged bisimilarity

We now define the *hedged bisimilarity*, firstly introduced in [3]. A *hedge* is a finite subset of $\mathcal{M} \times \mathcal{M}$ (where $\mathcal{M}$ is the set of all messages); we denote by $\mathcal{H}$ the set of all hedges. A hedge h expresses a relation between messages used by two processes P and Q, so that if $(M, N) \in h$ then M has the same effects on P that N has on Q. For this reason not all hedges are well-formed (e.g. a hedge that associates a name with a pair) and a notion of consistency is needed to understand which hedges are really relevant for the bisimulation.

Definition 3 (Consistent Hedge). *A hedge h is* consistent *if and only if it is* pair-free *(i.e. all messages in h are not pairs) and for $(M, N) \in h$ we have that:*

– $M \in \mathcal{N} \iff N \in \mathcal{N}$;
– *for all* $(M', N') \in h$, *if* $M \cong M'$ *or* $N \cong N'$ *then* $M = M'$ *and* $N = N'$;
– *if* $M \cong \{M'\}_k$ *e* $N \cong \{N'\}_j$ *then* $k \notin \pi_1(h)$ *and* $j \notin \pi_2(h)$.

Here (and afterward), we abuse the projection operators π_i to be used on any tuple, also outside of the calculus. The first condition requires a consistent hedge to match names with names. The second one requires that elements are taken up to equivalence on terms $\cong$. The last one requires that all encrypted messages cannot be decrypted using keys in h: encrypted messages in a consistent hedge cannot be reducible. Alongside hedges, we define *synthesis*, *analysis*, *irreducible*. The *synthesis* of a hedge h is the set of message pairs that can be built from h.

Definition 4 (Synthesis). *The synthesis* $S(h)$ *of a hedge* h *is the least set s.t.:*

– $h \subseteq S(h)$;
– *if* $(M, N) \in S(h)$, $(k, j) \in S(h)$ *and* $k, j \in \mathcal{N}$ *then* $(\{M\}_k, \{N\}_j) \in S(h)$;
– *if* $(M_1, N_1) \in S(h)$ *and* $(M_2, N_2) \in S(h)$ *then* $((M_1, M_2), (N_1, N_2)) \in S(h)$.

We write $h \vdash M \leftrightarrow N$ *for* $(M, N) \in S(h)$, *and in this case we say that* M *and* N *are* homologous *w.r.t.* h.

The *analysis* of a hedge is the set of all message pairs obtained by "opening" the messages of h via decryption or projection. The *irreducible* are those elements in the analysis of a hedge that cannot be reduced further. Formally:

Definition 5 (Analysis). *The analysis* $A(h)$ *of a hedge* h *is the least set s.t.:*

– $h \subseteq A(h)$;
– *if* $(\{M\}_k, \{N\}_j) \in A(h)$ *and* $(k, j) \in A(h)$ *then* $(M, N) \in A(h)$;
– *if* $((M_1, N_1), (M_2, N_2)) \in A(h)$ *then* $(M_1, M_2) \in A(h)$ *and* $(N_1, N_2) \in A(h)$.

Moreover, we define the irreducible $I(h)$ *of a hedge* h *as*

$$I(h) \triangleq A(h) \setminus (\{(C, D) \in A(h) \mid C \cong \{M\}_k,\ D \cong \{N\}_j,\ (k, j) \in A(h)\} \cup \{((M_1, N_1), (M_2, N_2)) \in A(h)\})$$

It should be noted that all elements that can be reduced in the analysis can be derived from the irreducible via synthesis, i.e. $S(I(h)) = S(A(h))$. Lastly, since every hedge is a finite set, its analysis and irreducible are also finite.

A *hedged relation* $\mathcal{R}$ is a subset of $\mathcal{H} \times \mathcal{P} \times \mathcal{P}$, where $\mathcal{P}$ is the set of all processes. We write $h \vdash P\mathcal{R}Q$ when $(h, P, Q) \in \mathcal{R}$. Moreover, we say that $\mathcal{R}$ is *consistent* if, for all $h \in \mathcal{H}$, $h \vdash P\mathcal{R}Q$ implies that the hedge h is consistent.

Definition 6 (Hedged simulation). *A consistent hedged relation* $\mathcal{R}$ *is a* hedged simulation *if, whenever* $h \vdash P\mathcal{R}Q$ *we have that:*

– *if* $P \xrightarrow{\tau} P'$ *then there exists* Q' *such that* $Q \Longrightarrow Q'$ *and* $h \vdash P'\mathcal{R}Q'$;
– *if* $P \xrightarrow{\overline{a}} (\nu \boldsymbol{m})\langle M\rangle P'$ *and* $\boldsymbol{m} \cap (\mathrm{fn}(P) \cup n(\pi_1(h))) = \emptyset$ *then there exist* $b \in \mathcal{N}$ *and a concretion* $(\nu \boldsymbol{n})\langle N\rangle Q'$ *such that* $h \vdash a \leftrightarrow b$, $\boldsymbol{n} \cap (\mathrm{fn}(Q) \cup n(\pi_2(h))) = \emptyset$, $Q \overset{\overline{b}}{\Longrightarrow} (\nu \boldsymbol{n})\langle N\rangle Q'$ *and* $I(h \cup \{(M, N)\}) \vdash P'\mathcal{R}Q'$;

– *if* $P \xrightarrow{a} (x)P'$ *then there exist* $b \in \mathcal{N}$ *and an abstraction* $(y)Q'$ *such that* $h \vdash a \leftrightarrow b$, $Q \stackrel{b}{\Longrightarrow} (y)Q'$ *and for all* $B \subset \mathcal{N}$ *finite such that* $B \cap (\mathrm{fn}(P) \cup \mathrm{fn}(Q) \cup n(h)) = \emptyset$ *and* $h \cup id_B$ *is consistent, for all pairs* (M, N) *of ground terms, if* $h \cup id_B \vdash M \leftrightarrow N$ *then* $h \cup id_B \vdash P'\{M/x\}\mathcal{R}Q'\{N/y\}$.

The first condition requires that for each τ-transition from P there is a path of τ-transitions from Q such that the two target processes are in the simulation $\mathcal{R}$. The second condition requires that for each output transition of P, labelled with $\overline{a}$, there is an output transition from Q labelled with $\overline{b}$ (and possibly preceded by some silent transitions); moreover, a and b are homologous in h and the processes after the two output operations are paired in $\mathcal{R}$ w.r.t. a consistent hedge that extends h by pairing the two messages M and N. The last condition requires that for each input transition of P with label a, there is an input transition from Q labelled with b (and possibly preceded by some silent transitions); moreover, a and b are homologous in h and for all finite set B of fresh names w.r.t. P, Q and h, the abstractions $(x)P'$ and $(x)Q'$ are paired in the simulation $\mathcal{R}$ for each input messages (M, N) homologous by $h \cup id_B$. Hedges simulation leads to the definition of hedged bisimulation and bisimilarity. Remarkably, hedged bisimilarity coincides with barbed equivalence [3].

Definition 7 (Hedged bisimulation/bisimilarity). *A hedged simulation* $\mathcal{R}$ *is a* hedged bisimulation *if* $\mathcal{R}^{-1} = \{(h^{-1}, Q, P) \mid h \vdash P\mathcal{R}Q\}$ *is also a hedged simulation (where* $h^{-1} = \{(N, M) | (M, N) \in h\}$*). The* hedged bisimilarity*, written* $\sim$*, is the greatest hedged bisimulation, i.e. the union of all hedged bisimulations.*

4 Decidability of hedged bisimulation for finite processes

Clearly, bisimilarity is undecidable for general processes (see [5]). Hence, we focus on *finite* ones, i.e. without the "!" operator. Even on finite processes decidability of hedged bisimilarity is not obvious, since the third condition in Definition 6 requires to check the equivalence of two abstractions for an infinite number of messages w.r.t. any finite set of fresh names. In this section we show that this problem can be avoided as there is only a finite number of names and messages that need to be considered in deciding if two processes are hedged bisimilar.

The idea behind our result, and similar to that in [5] for framed bisimilarity, is the following: if $(x)P$ is finite, then it can inspect a message (using *let* and *guard* operators) up to a certain depth k. If a message M with more than k nested constructors is received by $(x)P$, then it can only be partially analysed by P. Hence, all messages M' equivalent to M up to depth k will not cause any difference in the execution of $(x)P$, apart from output messages. Indeed, $P\{M/x\}$ and $P\{M'/x\}$ can output different messages (i.e. different parts of M and M' respectively), but the two outputs are derived from M and M' by applying the same operations, and only messages obtained through decryption are interesting, since they can update the hedge h yielding a richer theory.

Before formalizing this idea, we introduce a canonical form of processes and show that any process can be translated into an equivalent one in canonical form.

Definition 8 (Canonical form). *A process P is in* canonical form *if in P*
- *any $[t_1 = t_2]$ is such that t_1 and t_2 are variables or messages;*
- *constructors do not occur inside terms t of* `let` $x = t$ `in` Q *operators;*
- *destructors do not occur inside any term t of output operators $\overline{a}\langle t\rangle Q$;*
- *for any occurrence of isname(t), ispair(t) and isenc(t) in P, t is a variable.*

Proposition 1. *For all P there is a canonical process Q such that $P \equiv Q$.*

The proof is based on defining a rewriting system on processes which preserves congruence and terminates on canonical processes. Thanks to this result, we can restrict ourselves to processes in canonical form without loss of generality.

For terms and boolean expressions, we define their *maximal constructor depth*. Intuitively, this depth is related to the maximum size of messages that can satisfy a boolean expression when replacing a free variable appearing in it.

Definition 9 (Maximal constructor depth). *The* maximal constructor depth $\mathrm{mcd}(t)$ *of a term t is defined inductively by the clauses*

$$\mathrm{mcd}(n) = 0 \qquad \mathrm{mcd}((t_1, t_2)) = \max(\mathrm{mcd}(t_1), \mathrm{mcd}(t_2)) + 1$$
$$\mathrm{mcd}(x) = 0 \qquad \mathrm{mcd}(\{t_1\}_{t_2}) = \mathrm{mcd}(t_1) + 1$$

and then extended to boolean formulas as follows:

$$\mathrm{mcd}(\mathbf{true}) = 0 \qquad \mathrm{mcd}(\phi_1 \wedge \phi_2) = \max(\mathrm{mcd}(\phi_1), \mathrm{mcd}(\phi_2))$$
$$\mathrm{mcd}(\neg\phi) = \mathrm{mcd}(\phi) \qquad \mathrm{mcd}([x = M]) = \mathrm{mcd}(M)$$
$$\mathrm{mcd}([x = y]) = 0 \qquad \mathrm{mcd}(\mathit{isname}(x)) = 1$$
$$\mathrm{mcd}(\mathit{ispair}(x)) = 1 \qquad \mathrm{mcd}(\mathit{ispair}(x)) = 1$$

Definition 10 (k-homologous). *Given $h \in \mathcal{H}$ and $M, N \in \mathcal{M}$, we define*

$$h \vdash_k M \leftrightarrow N \overset{\triangle}{\iff} h \vdash M \leftrightarrow N \text{ and } k = \max(\mathrm{mcd}(M), \mathrm{mcd}(N))$$

Whenever $h \vdash_k M \leftrightarrow N$ we say that M and N are k-homologous *in h.*

The notion of k-homologous terms allows us to deduce an upper bound on the number of names that are required to prove that $h \vdash M \leftrightarrow N$.

Lemma 1. *Let $h \in \mathcal{H}$ and $M, N \in \mathcal{M}$ be such that $\max(\mathrm{mcd}(M), \mathrm{mcd}(N)) = k$. If there is a finite set of names $B \subset \mathcal{N}$ such that $B \cap n(h) = \emptyset$, $h \cup id_B$ is consistent and $h \cup id_B \vdash_k M \leftrightarrow N$, then there exists $B' \subset \mathcal{N}$ such that $|B'| \leq 2^k$ and satisfying the same three properties.*

Proof. Suppose $|B| \geq 2^k$ and that $h \vdash M \leftrightarrow N$ does not hold, otherwise the lemma is trivially satisfied. We have that M and N are in the synthesis $S(h \cup id_B)$ and, at worst, all names in M and N are in B. As $k = \max(\mathrm{mcd}(M), \mathrm{mcd}(N))$ and both *encrypt* and *pairing* are binary constructors, M and N can be represented as binary trees with height k and with therefore at most 2^k leafs. Hence B can be reduced to a set B' such that $|B'| \leq 2^k$ and $h \cup id_{B'} \vdash_k M \leftrightarrow N$. As $B' \subseteq B$, it also holds that $h \cup id_{B'}$ is consistent and $B' \cap n(h) = \emptyset$. □

Lemma 1 leads to the notion of *d-hedged bisimilarity*: a hedged bisimilarity where size of messages and number of fresh names are bounded. We define first the corresponding simulation; differences with Definition 6 are put in boxes.

Definition 11 (d-hedged simulation). *For any integer $d \geq 0$, a consistent hedged relation $\mathcal{R}$ is a* d-hedged simulation *if whenever $h \vdash P\mathcal{R}Q$ we have that:*

- *if $P \xrightarrow{\tau} P'$ then there exists Q' such that $Q \Longrightarrow Q'$ and $h \vdash P'\mathcal{R}Q'$;*
- *if $P \xrightarrow{\overline{a}} (\nu \boldsymbol{m})\langle M\rangle P'$ and $\boldsymbol{m} \cap (\mathrm{fn}(P) \cup n(\pi_1(h))) = \emptyset$ then there exist $b \in \mathcal{N}$ and a concretion $(\nu \boldsymbol{n})\langle N\rangle Q'$ such that $h \vdash a \leftrightarrow b$, $\boldsymbol{n} \cap (\mathrm{fn}(Q) \cup n(\pi_2(h))) = \emptyset$, $Q \overset{\overline{b}}{\Longrightarrow} (\nu \boldsymbol{n})\langle N\rangle Q'$ and $I(h \cup \{(M, N)\}) \vdash P'\mathcal{R}Q'$;*
- *if $P \xrightarrow{a} (x)P'$ then there exist $b \in \mathcal{N}$ and an abstraction $(y)Q'$ such that $h \vdash a \leftrightarrow b$, $Q \overset{b}{\Longrightarrow} (y)Q'$ and for all $B \subset \mathcal{N}$, where $\boxed{|B| \leq 2^d}$, $B \cap (\mathrm{fn}(P) \cup \mathrm{fn}(Q) \cup n(h)) = \emptyset$ and $h \cup id_B$ is consistent, for all pairs (M, N) of ground terms, if $\boxed{\exists k \leq d\ h \cup id_B \vdash_k M \leftrightarrow N}$ then $h \cup id_B \vdash P'\{M/x\}\mathcal{R}Q'\{N/y\}$.*

Definition 12 (d-hedged bisimulation/bisimilarity). *A* d-hedged bisimulation *is a d-hedged simulation $\mathcal{R}$ such that $\mathcal{R}^{-1} = \{(h^{-1}, Q, P) \mid h \vdash P\mathcal{R}Q\}$ is also a d-hedged simulation (where $h^{-1} = \{(N, M) | (M, N) \in h\}$). The* d-hedged bisimilarity*, written $\sim^d$, is the greatest d-hedged bisimulation, i.e. the union of all d-hedged bisimulations.*

Since a d-hedged bisimulation is a hedged bisimulation up to a certain bound on the size of messages and fresh names, its definition leads to the following result.

Lemma 2. *(1) Any hedged bisimulation is a d-hedged bisimulation for all $d \geq 0$. (2) When $d > 0$, a d-hedged bisimulation is also a $(d-1)$-hedged bisimulation.*

We now aim to show that for any processes P, Q, there exists $d \geq 0$ such that $\exists h \in \mathcal{H}\ h \vdash P \sim^d Q \Rightarrow \exists h \in \mathcal{H}\ h \vdash P \sim Q$. This statement is not valid for arbitrary infinite processes since these can analyse messages of arbitrary depth. Therefore, we now consider only the fragment of spi-calculus without replication.

Notice that *let* and *guard* operators are the only constructs that can check the structure of messages. For instance, $c(y)(\texttt{let}\ x = \pi_1(\mathtt{D}_b(y))\ \texttt{in}\ [x = t].P)$ first decompose the message received on the channel c by applying $\mathtt{D}_b(y)$ and π_1, and then test the result against a term t. Therefore, messages with constructor depth greater than $|t| + 2$ (where 2 refers to the number of destructors in the let expression) will automatically fail the test. For `let` expressions, this observation leads to the following definition of *analysis depth*.

Definition 13 (Analysis depth). *Let P be a finite (canonical) process. The* analysis depth *of P, denoted by* $\mathrm{ad}(P)$*, is defined inductively by the clauses*

$$\mathrm{ad}(0) = 0 \qquad \mathrm{ad}(P|Q) = \mathrm{ad}(P) + \mathrm{ad}(Q)$$
$$\mathrm{ad}((\nu \boldsymbol{m})P) = \mathrm{ad}(P) \qquad \mathrm{ad}(\phi.P) = \mathrm{ad}(P)$$
$$\mathrm{ad}(\overline{M}\langle t\rangle.P) = \mathrm{ad}(P) \qquad \mathrm{ad}(P + Q) = \max(\mathrm{ad}(P), \mathrm{ad}(Q))$$
$$\mathrm{ad}(M(x).P) = \mathrm{ad}(P) \qquad \mathrm{ad}(\texttt{let}\ x = t\ \texttt{in}\ P) = \mathrm{ad}(P) + \mathrm{mdd}(t)$$

where the maximal destructor depth mdd *of a term is defined as follows:*

$$\begin{aligned} \mathrm{mdd}(n) &= 0 & \mathrm{mdd}(\{t_1\}_{t_2}) &= \max(\mathrm{mdd}(t_1), \mathrm{mdd}(t_2)) \\ \mathrm{mdd}(x) &= 0 & \mathrm{mdd}((t_1, t_2)) &= \max(\mathrm{mdd}(t_1), \mathrm{mdd}(t_2)) \\ \mathrm{mdd}(\pi_i(t)) &= \mathrm{mdd}(t) + 1 & \mathrm{mdd}(\mathtt{D}_{t_2}(t_1)) &= \max(\mathrm{mdd}(t_1), \mathrm{mdd}(t_2)) + 1. \end{aligned}$$

Analysis depth, together with the maximal constructor depth previously defined, are used to derive the upper-bound on the size of messages that are needed to test in order to correctly decide if two processes are hedged bisimilar. We extend the notion of maximal constructor depth to hedges and processes, where the latter is well-defined as our processes are finite. For any hedge h and process P,

$$\begin{aligned} \mathrm{mcd}(h) &= \min\{k \mid \text{for all } (M,N) \in h : h \vdash_k M \leftrightarrow N\} \\ \mathrm{mcd}(P) &= \max\{\mathrm{mcd}(\phi) \mid P(\equiv \cup \Longrightarrow)Q \text{ and } \phi \text{ occurs in } Q\} \end{aligned}$$

Definition 14 (Critical depth). *Let P and Q be two finite processes and let h be a hedge. The* critical depth $\mathrm{CD}(h, P, Q)$ *is defined by*

$$\mathrm{CD}(h, P, Q) \triangleq \mathrm{mcd}(h) + \max(\mathrm{ad}(P) + \mathrm{mcd}(P),\ \mathrm{ad}(Q) + \mathrm{mcd}(Q))$$

As we will formally see below, when checking if a d-hedged simulation exists, given a hedge h we can correlate two input $P \xrightarrow{a} (x)P'$ and $Q \xrightarrow{b} (x)Q'$, simply by testing the equivalence between P' and Q' w.r.t. messages with *maximal constructor depth* less or equal than $\mathrm{CD}(h, P, Q)$. Moreover, Lemma 1 limits the number of fresh names we must consider to $2^{\mathrm{CD}(h,P,Q)}$. Messages that exceed these bounds can indeed be pruned without affecting the behaviour of processes.

Definition 15 (d-pruning). *Let $M, N \in \mathcal{M}$ and h be a consistent hedge such that $h \vdash M \leftrightarrow N$. For $d \geq 0$, the* d-pruning *of M, N w.r.t. h, denoted by $\mathrm{pr}_d(h, M, N)$, is defined as (h, M, N) if $(M, N) \in h$, and otherwise as:*

$$\begin{aligned} \mathrm{pr}_0(h, M, N) &= (h \cup id_{\{a\}}, a, a) \text{ where } a \in \mathcal{N} \text{ is fresh} \\ \mathrm{pr}_{d+1}(h, \{U\}_J, \{V\}_K) &= (h', \{M'\}_J, \{N'\}_K) \\ &\quad \text{where } (J,K) \in h \text{ and } \mathrm{pr}_d(h, U, V) = (h', M', N') \\ \mathrm{pr}_{d+1}(h, (L_1, R_1), (L_2, R_2)) &= (h'', (L_1', R_1'), (L_2', R_2')) \\ &\quad \text{where } \mathrm{pr}_d(h, L_1, L_2) = (h', L_1', L_2') \\ &\quad \text{and } \mathrm{pr}_d(h', R_1, R_2) = (h'', R_1', R_2'). \end{aligned}$$

Intuitively, the d-pruning of a message pair (M, N) generates a message pair (M', N') where subterms appearing at levels greater than d are replaced by fresh names w.r.t. h, M and N. Then, the d-pruning is unique up to fresh names. Critical depth and d-pruning are readily extended to single processes and messages as $\mathrm{CD}(h, P) \triangleq \mathrm{CD}(h, P, P)$ and $\mathrm{pr}_d(h, M) \triangleq \mathrm{pr}_d(h, M, M)$. As stated in the next two lemmata, pruning terms that exceeds $\mathrm{CD}(h, P)$ do not modify the behaviour of the process P.

Lemma 3. *Let $(x)P$ be an abstraction of a finite process, h be a hedge and $d \geq \mathrm{CD}(h, P)$. For every message M the pruning $N = \pi_2(\mathrm{pr}_d(h, M))$ does not affect the reduction relation. That is,*

- $(\phi.P)\{M/x\} > P\{M/x\}$ *if and only if* $(\phi.P)\{N/x\} > P\{N/x\}$;
- $(\mathtt{let}\ y = t\ \mathtt{in}\ P)\{M/x\} > P\{M/x\}\{[\![t\{M/y\}]\!]\}$ *if and only if* $(\mathtt{let}\ y = t\ \mathtt{in}\ P)\{N/x\} > P\{N/x\}\{[\![t\{N/y\}]\!]\}$

Lemma 4. *Let $(x)P$ be an abstraction of a finite process, h be a hedge and $d \geq \mathrm{CD}(h, P)$. For every message M the pruning $N = \pi_2(\mathrm{pr}_d(h, M))$ is so that*

- $P\{M/x\} \xrightarrow{n} (y)P'\{M/x\}$ *if and only if* $P\{N/x\} \xrightarrow{n} (y)P'\{N/x\}$;
- $Q\{M/x\} \xrightarrow{\overline{n}} ((\nu \boldsymbol{m})\langle t\rangle Q')\{M/x\}$ *iff* $Q\{N/x\} \xrightarrow{\overline{n}} ((\nu \boldsymbol{m})\langle t\rangle Q')\{N/x\}$.

This lemma also implies that τ-transitions are preserved, because τ-transitions are introduced only by the interaction rule (Figure 3); as shown in the diagram below, the conclusions of this rule are preserved because the premises are.

$$\left(\text{Lemma 4}\right) \quad \frac{P\{M/x\} \xrightarrow{n} ((y)P')\{M/x\} \quad Q\{M/x\} \xrightarrow{\overline{n}} ((\nu \boldsymbol{m})\langle t\rangle Q')\{M/x\}}{(P|Q)\{M/x\} \xrightarrow{\tau} ((\nu \boldsymbol{m})(P'\{t/y\}|Q'))\{M/x\}} \quad \left(\text{Lemma 4}\right)$$

$$\frac{P\{N/x\} \xrightarrow{n} ((y)P')\{N/x\} \quad Q\{N/x\} \xrightarrow{\overline{n}} ((\nu \boldsymbol{m})\langle t\rangle Q')\{N/x\}}{(P|Q)\{N/x\} \xrightarrow{\tau} ((\nu \boldsymbol{m})(P'\{t/y\}|Q'))\{N/x\}}$$

We can then reduce hedged bisimulation to d-hedged bisimulation.

Theorem 1. *Let P and Q be two finite processes, and $h \in \mathcal{H}$ a consistent hedge. There is an integer d such that $h \vdash P \sim^d Q$, if and only if $h \vdash P \sim Q$.*

Proof. ($\Leftarrow$) A hedged bisimulation is a d-bisimulation for all d.
($\Rightarrow$) By Lemmata 3, 4 and the fact that the following is a hedged bisimulation:

$$\left\{ (h, P, Q) \;\middle|\; \begin{array}{l} P, Q \text{ finite and } \exists h' \in \mathcal{H}\ \exists P', Q' \in \mathcal{P}\ \exists M, N \in \mathcal{M} \text{ s.t.} \\ P = P'\{M/x\}, Q = Q'\{N/y\}, (h', M', N') = \mathrm{pr}_d(h, M, N), \\ h' \vdash P'\{M'/x\} \sim^d Q'\{N'/y\}, \text{for } d = \mathrm{CD}(h, P, Q) \end{array} \right\} \quad \square$$

As every quantification is bounded, d-hedged bisimilarity is decidable on finite processes. Then, so is hedged bisimilarity.

Corollary 1. *Let P and Q be two finite processes and $h \in \mathcal{H}$ a consistent hedge. It is decidable whether $h \vdash P \sim Q$.*

An algorithm to decide hedged bisimilarity is shown in Figure 4. For P, Q two finite (canonical) processes and h a hedge (which represents the initial knowledge of the attacker, e.g. public channels, known keys, nonces, etc.), $\mathcal{HB}(h, P, Q) =$ **true** if and only if there is a d such that (h, P, Q) are d-hedged bisimilar. Hence, by Theorem 1 we have $h \vdash P \sim Q$ if and only if $\mathcal{HB}(h, P, Q) =$ **true**.

$\mathcal{HB}(h,P,Q) = H(h,P',Q') \wedge H(h^{-1},Q',P')$

$H(h,P,Q) =$

for each $P \xrightarrow{\tau} P'$ **select** $Q \Longrightarrow Q'$ **such that** $\mathcal{HB}(h,P',Q')$

for each $P \xrightarrow{\overline{a}} (\nu \boldsymbol{m})\langle t\rangle P'$ **select** $Q \overset{\overline{b}}{\Longrightarrow} (\nu \boldsymbol{n})\langle t'\rangle Q'$ **such that**

$(a,b) \in I(h)$, $h' := I(h \cup \{([\![t]\!],[\![t']\!])\})$ consistent **and** $\mathcal{HB}(h',P',Q')$

for each $P \xrightarrow{a} (x)P'$

let $d = \mathrm{CD}(h,P,Q)$ **and select** $(Q \overset{b}{\Longrightarrow} (x)Q'$ **and** $B \subset \mathcal{N})$ **such that**

$(a,b) \in I(h)$, $|B| = 2^d$, $B \cap (\mathrm{fn}(P) \cup \mathrm{fn}(Q) \cup n(h) \cup \{a,b\}) = \emptyset$, $h' := h \cup id_B$ consistent

and for each (M,N) **s.t.** $(\exists k \leq d :\ h' \vdash_k M \leftrightarrow N) : \mathcal{HB}(h', P'\{M/x\}, Q'\{N/x\})$

Fig. 4. Algorithm to decide hedged bisimilarity. The **select** statement implements a non-deterministic exploration of the (finite) possible choices of its argument, until the condition is satisfied; it returns **true** if successful, **false** otherwise.

5 Conclusions and further work

In this paper we have proved that hedged bisimilarity (and hence barbed equivalence) is decidable on finite processes of the spi-calculus. As we generalised the spi-calculus to abstract from the syntactical equality between terms, our algorithm can be readily applied to different encryption/decryption schemata just by changing the congruence rules, as long as some mild conditions are satisfied.

Thanks to this extension, we believe that the approach followed in this paper can be applied to decide hedged bisimilarity between processes of more expressive calculi, in particular the *applied π-calculus* (which is used in tools such as ProVerif [2]) and calculi with quantitative aspects [6,4]. Also considering larger fragments of the spi-calculus beyond finite processes, e.g. *depth-* and *restriction-bounded* processes, seems particularly promising.

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.
2. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *Proc. 20th LICS*, pages 331–340. IEEE, 2005.
3. J. Borgström and U. Nestmann. On bisimulations for the spi calculus. *Mathematical Structures in Computer Science*, 15(3):487–552, 2005.
4. T. Brengos, M. Miculan, and M. Peressotti. Behavioural equivalences for coalgebras with unobservable moves. *J. Logica. Algebraic Methods Program.*, 84:826–852, 2015.
5. H. Hüttel. Deciding framed bisimilarity. In *Proceedings of Infinity'02*, volume 68 of *Electronic Notes in Theoretical Computer Science*, pages 1–18, 2003.
6. M. Miculan and M. Peressotti. GSOS for non-deterministic processes with quantitative aspects. In *Proc. QAPL*, volume 154 of *ENTCS*, pages 17–33, 2014.
7. R. Milner and D. Sangiorgi. Barbed bisimulation. In *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer, 1992.
8. D. Sangiorgi, D. Walker. *The π-calculus: a Theory of Mobile Processes.* CUP, 2001.