Encoding cryptographic primitives in a calculus with polyadic synchronisation

Joana Martinho
Department of Mathematics, Instituto Superior Técnico, Technical University of Lisbon, Portugal
joana_dk@portugalmail.pt
António Ravara
Security and Quantum Information Group, Instituto de Telecomunicações, and
Department of Mathematics, Instituto Superior Técnico, Technical University of Lisbon, Portugal
aravara@ist.utl.pt

Abstract

We thoroughly study the behavioural theory of epi**, a π-calculus extended with polyadic synchronisation. We show that the natural contextual equivalence, barbed congruence, coincides with early bisimilarity, which is thus its co-inductive characterisation. Moreover, we relate early bisimilarity with the other usual notions, ground, late and open, obtaining a lattice of equivalence relations that clarifies the relationship among the "standard" bisimilarities.**
Furthermore, we apply the theory developed to obtain an expressiveness result: epi **extended with key encryption primitives may be fully abstractly encoded in the original** epi **calculus. The proposed encoding is sound and complete with respect to barbed congruence; hence, cryptographic** epi (crypto-epi) **gets behavioural theory for free, what contrasts with other process languages with cryptographic constructs that usually require a big effort to develop such theory.**
Therefore, it is thus possible to use crypto-epi **to analyse and to verify properties of security protocols using equational reasoning. To illustrate this claim, we prove the symmetric and asymmetric cryptographic system laws, and the correctness of a protocol of secure message exchange.**

ACM classification

D.2.4 Program Verification: Correctness Proofs.

F.1.1 Models of Computation: Process Algebra.

F.1.3 Specification, Verification, and Reasoning about Programs: Logics and Meanings of Programs.

F.3.3 Program constructs: Control Primitives.

General terms: Behavioural theory, Cryptographic Systems, Mobile Calculus

Keywords: Barbed Congruence, (Early) Bisimulation, Cryptographic Primitives, Fully Abstract Encoding, π-calculus, Polyadic Synchronisation

Contents

Note. Joana Martinho was affiliated with the Department of Mathematics, Instituto Superior Técnico, when developing a preliminary version of this work. António Ravara is the corresponding author.

1. INTRODUCTION

We study herein the behavioural theory of a π-calculus where, instead of on simple names, processes synchronise in vectors of names (epi, acronym standing for *extended* π-calculus). To illustrate the expressive power of the calculus and a possible application area, we show that some cryptographic primitives are derivable in epi. Therefore, if one adds such primitives to the calculus, the resulting process language enjoys the same theory of the original language, and thus one may use it to prove properties of security protocols.

Extended π-calculus. The π-calculus with polyadic synchronisation (epi), proposed by Carbone and Maffeis [CM03] is an extension of the π-calculus of Milner, Parrow, and Walker [MPW92, SW01] that generalises the synchronisation mechanism, based on handshaking, *i.e.*, the simultaneous execution of input/output actions, by allowing channel names to be composite.

The fact that in epi communication is only established if the channel vectors match element-wise, enhances its expressive power with respect to the π-calculus. In particular, Carbone and Maffeis show that the

matching construct[1] can be encoded in the π-calculus with polyadic synchronisation but not in the π-calculus. In addition, they also prove that the higher the degree of synchronisation (*i.e.* the maximum length of the channel vectors), the higher the expressive power of the calculus.

Carbone and Maffeis did not fully develop the behavioural theory of the process language they proposed. Defining a grammar and an operational semantics yields a description language and a rigorous definition of its computational behaviour, but a calculus (in the logical sense) requires a theory to equate terms. A process *calculus* is achieved either by axiomatically, inductively, or co-inductively defining a behavioural equivalence (ideally a congruence).

Goals and contributions. The aim of this paper is twofold: to develop the behavioural theory of epi, defining a contextual equivalence and looking for its co-inductive characterisation; and to use this theory to show how to define cryptographic primitives preserving it, thus allowing the calculus extended with such primitives to be used to analyse and to verify security protocols.

The first goal is hence to study in detail the behavioural semantics of epi: (1) defining an operational semantics (a late labelled transition system semantics); (2) defining the usual equivalence notions (in the context of mobile calculi): contextual and co-inductive (ground, late, early and open bisimilarities); and (3) extending results from the π-calculus to epi, namely obtaining congruence results, finding which (if any) bisimilarity coincides with the contextual equivalence, and establishing a lattice of inter-relations between the various equivalence relations. We find that, in epi, like in π, barbed congruence coincides with early congruence (early bisimilarity closed for all substitutions), and thus, we have a co-inductive characterisation of the "natural" contextual equivalence of the calculus. Moreover, we relate all these "standard" notions of co-inductive equivalences, ground, late, early, and open, bisimilarities and congruences, obtaining a lattice of equivalence relations that clarifies the relationship among them.[2] To our knowledge, this is original work, and provides to epi the basic behavioural theory of a mobile calculus.

The second goal is to use the theory developed to show that epi extended with key encryption primitives may be used to analyse and to verify security protocols. To explore this possible application area, and to further show the expressive power of epi, we define a new calculus, the cryptographic π-calculus with polyadic synchronisation (crypto-epi), an extension of epi with the referred cryptographic primitives (in the spirit of the spi-calculus of Abadi and Gordon [AG97] — itself an extension of the π-calculus with constructs that allow for encryption and decryption of messages — or of the applied π-calculus of Abadi and Fournet [AF01] — another extension of the π-calculus with a term algebra). These two primitives are suggested by Carbone and Maffeis in the introduction of their paper [CM03] as another argument to support the expressiveness of the calculus they propose — epi. As a first contribution, we prove, using those primitives, the symmetric and asymmetric cryptographic system laws. Since Carbone and Maffeis did not define nor study the extension of the calculus with such primitives, herein, we formally define crypto-epi: (1) adding to the grammar of epi primitives for (symmetrically) encoding and decoding names; (2) providing transition rules to deal with these constructs, enriching the labelled transition system of epi; (3) extending results from epi to crypto-epi, namely showing that the new constructs preserve early bisimilarity. Then we show that crypto-epi is fully abstractly encoded, with respect to barbed congruence, in the original π-calculus with polyadic synchronisation, thus reflecting the behavioural theory of epi back to crypto-epi, and allowing the usual reasoning principles using behavioural equivalences to be used in the latter.

The encoding is also proposed by Carbone and Maffeis in the introduction of their paper, but they do not study its properties. To our knowledge, our result is original: not only it shows that these cryptographic primitives may be defined in epi as programming constructs and do not need to be primitive, re-enforcing the expressive power of epi, but also it provides standard behavioural theory to a cryptographic mobile process calculus. Moreover, since the results closely follow those of the pi-calculus, it should be straightforward to adapt tools like the Mobility Workbench [Vic94, VM94] to epi and crypto-epi, achieving a powerful tool to prove by equational reasoning properties of security protocols. Note that other cryptographic calculi like the spi-calculus or the applied pi-calculus have a more evolved and sometimes cumbersome

[1] The matching construct is the process **if** $x = y$ **then** P; it compares names x and y and, if they coincide, behaves like process P; otherwise does nothing. It is fully abstractly encoded in epi (assuming no matching in P) as the process $(\nu z)(\overline{z \cdot x}\langle\rangle \mid z \cdot y().P)$.
[2] The result is a lattice similar to that of the π-calculus.

P ::=	processes	π ::=	prefixes
$\mathbf{0}$	*inaction*	τ	*internal action*
$\mid \pi.P$	*prefix*	$\mid x_1 \cdot ... \cdot x_k(y)$	*input*
$\mid !P$	*replication*	$\mid \overline{x_1 \cdot ... \cdot x_k}\langle y \rangle$	*output*
$\mid (\nu x)P$	*restriction*		
$\mid (P \mid P)$	*parallel composition*		
$\mid (P + P)$	*choice*		

FIGURE 1: crypto-epi syntax

behavioural theory. The extra structure for data handling severely complicates equational reasoning: naïve adaptation of bisimulations are not adequate; new notions developed are "heavy", and difficult to automate [AF01, AG97, BAF07, BNP02, BN05]. To illustrate the use of the theory developed, we prove the correctness of a protocol of secure message exchange.

Structure of the paper. We structure the presentation of our work in the following manner:

- In Section 2 we introduce the syntax and a late labelled transition semantics of the π-calculus with polyadic synchronisation, epi, as first proposed by Carbone and Maffeis.
- In Section 3 we define the four usual co-inductive notions of equivalence (ground, late, early and open bisimilarity), and compare these notions, concluding that they relate to each other just as in the π-calculus. We further introduce the notions of barbed bisimilarity, equivalence and congruence, and conclude the latter coincides with early congruence. Although relying on a similar result obtained for the π-calculus [San92], the proof of the coincidence of the notions in epi requires several adjustments.
- In Section 4 we extend epi with the cryptographic primitives proposed by Carbone and Maffeis [CM03]. In addition to their work, we give an operational semantics to the new calculus, adding new rules to the original labelled transition system, and moreover, we analyse in detail a simple cryptographic protocol, proving it correct.
- In Section 5 we prove fully abstract (with respect to barbed congruence) the encoding of the cryptographic constructs in epi.
- Section 6 concludes the paper, listing our contributions and giving directions for future research.
- The proofs that are technically more elaborate are presented in appendices.

2. epi: π-CALCULUS WITH POLYADIC SYNCHRONISATION

The π-calculus with polyadic synchronisation, epi, is a variant of the π-calculus where the channels can consist of sequences of names and communication is established if and only if the channel vectors match element-wise.[3]

2.1. Syntax

We introduce the syntax of the calculus in detail and also mention some of the main differences between this and the π-calculus. These differences will be explained in further detail in subsequent parts throughout this section.

Definition 2.1 *Processes*
Let N be a countable set of names and $x, x_1, ..., x_k, y$ range over N for some $k \in \mathbb{N}$. The grammar in Figure 1 defines the class of processes *$\mathcal{P}_S$, ranged over by P, Q.*

The decreasing order of precedence of operators follows that of the definition, where the prefix operator has the highest precedence. In what follows we use the notation π for $\pi.\mathbf{0}$, and $(\nu z, w)P$ for $(\nu z)(\nu w)P$.

[3] We call π-calculus with biadic synchronisation the particular case of the π-calculus with polyadic synchronisation where the composite channels have at most two names.

Action	Description	$\mathsf{fn}(\alpha)$	$\mathsf{bn}(\alpha)$	$\mathsf{nm}(\alpha)$
τ	internal action	$\emptyset$	$\emptyset$	$\emptyset$
$\overline{u}\langle y\rangle$	free output	$\mathsf{nm}(u) \cup \{y\}$	$\emptyset$	$\mathsf{nm}(u) \cup \{y\}$
$\overline{u}(y)$	bound output	$\mathsf{nm}(u)$	$\{y\}$	$\mathsf{nm}(u) \cup \{y\}$
$u(y)$	input	$\mathsf{nm}(u)$	$\{y\}$	$\mathsf{nm}(u) \cup \{y\}$

TABLE 1: Actions

Process	Description	$\mathsf{fn}(P)$	$\mathsf{bn}(P)$	$\mathsf{nm}(P)$
$\mathbf{0}$	inaction	$\emptyset$	$\emptyset$	$\emptyset$
$\pi.Q$	prefix	$\mathsf{fn}(\pi) \cup (\mathsf{fn}(Q) \setminus \mathsf{bn}(\pi))$	$\mathsf{bn}(\pi) \cup \mathsf{bn}(Q)$	$\mathsf{nm}(\pi) \cup \mathsf{nm}(Q)$
$!Q$	replication	$\mathsf{fn}(Q)$	$\mathsf{bn}(Q)$	$\mathsf{nm}(Q)$
$(\nu y)Q$	restriction	$\mathsf{fn}(Q) \setminus \{y\}$	$\{y\} \cup \mathsf{bn}(Q)$	$\{y\} \cup \mathsf{nm}(Q)$
$(Q_1 \mid Q_2)$	parallel composition	$\mathsf{fn}(Q_1, Q_2)$	$\mathsf{bn}(Q_1, Q_2)$	$\mathsf{nm}(Q_1, Q_2)$
$(Q_1 + Q_2)$	choice	$\mathsf{fn}(Q_1, Q_2)$	$\mathsf{bn}(Q_1, Q_2)$	$\mathsf{nm}(Q_1, Q_2)$

TABLE 2: Names in Processes

All operators used here are also present in the π-calculus and their behavior is as expected. Nonetheless, note that restriction is made on names as in the π-calculus and not on composite channels: this allows for *partial restriction*.

One should also note that in the π-calculus with polyadic synchronization it is not necessary to include the match operator since it can be encoded in the calculus. This is not possible in a 'sensible' manner using the original π-calculus that, therefore, takes the match operator as a primitive. This important separation result between the two calculi was obtained by Carbone and Maffeis [CM03], and it is the central expressiveness result about epi.

Consider $u = x_1 \cdot ... \cdot x_k$ and $\overline{u} = \overline{x_1 \cdot ... \cdot x_k}$, where $k \in \mathbb{N}$, represent respectively the input and output channel vectors. Then, $\mathsf{nm}(u) = \mathsf{nm}(\overline{u}) = \{x_1, ..., x_k\}$. As in the π-calculus, there are four possible kinds of *actions* α in the present calculus, as seen in Table 1. Let $\mathsf{bn}(\alpha)$ denote the set of *bound names* in α, $\mathsf{fn}(\alpha)$ the set of *free names* in α and $\mathsf{nm}(\alpha)$ the set of all *names* in α (the union of the previous two sets). The respective notions for prefixes, *i.e.*, $\mathsf{fn}(\pi)$, $\mathsf{bn}(\pi)$, and $\mathsf{nm}(\pi)$, are defined similarly. Furthermore, the notions of bound and free names in a process P, denoted by $\mathsf{bn}(P)$ and $\mathsf{fn}(P)$ respectively, follow from those of the π-calculus. Table 2 presents the rigorous definition of these notions, where $\mathsf{nm}(P)$ denotes the names in the process P. Let $\mathsf{fn}(P_1, P_2) = \mathsf{fn}(P_1) \cup \mathsf{fn}(P_2)$, and consider similar definitions for $\mathsf{bn}(P_1, P_2)$ and $\mathsf{nm}(P_1, P_2)$.

Note that we sometimes use polyadic CCS-like prefixes $\overline{a \cdot w}$ and $a \cdot y$ where no item is being sent or expected to be received. We do this to highlight the fact that what could be transmitted is irrelevant, the problem lies in the synchronization of the composite channels. In general, $\overline{u}.P$ will be used as shorthand for $\overline{u}\langle y\rangle .P$ for some y, and $u.P$ will be used as shorthand for $u(y).P$ where $y \notin \mathsf{fn}(P)$.

Substitution and α-convertibility are defined as in the π-calculus [SW01], though we now require that the latter takes into account the possibility of composite channels. For the sake of clarity, we provide the formal definitions next.

Definition 2.2 *Substitution*
Let $w \notin \mathsf{bn}(P)$ where $P \in \mathcal{P}_S$.

1. *The result of applying the* substitution $\sigma = \{w/z\}$ *to process P, written $P\sigma$ or $P\{w/z\}$, is the process obtained by replacing each free occurrence of z in P by w.*
2. *The result of applying the* simultaneous substitution $\sigma = \{w_1, ..., w_n/x_1, ..., x_n\}$*, for distinct x_i, to process P where $w_1, ..., w_n \notin \mathsf{bn}(P)$, also written $P\sigma$, is the process obtained by simultaneously replacing each free occurrence of x_i in P by w_i where $1 \leq i \leq n$ and $n \in \mathbb{N}$.*

Note that given a substitution $\sigma = \{w/z\}$ we denote the result of applying σ to z as $\sigma(z)$. In this case, we then have that $\sigma(z) = w$. Moreover, substitution may imply the renaming via α-conversion of bound actions to avoid unwanted captures of free names. The definition of this notion follows.

Definition 2.3 *α-convertibility*
Let $u = x_1 \cdot ... \cdot x_k$ where $k \in \mathbb{N}$.

1. *A* change of bound names *in a process P is the replacement of a subterm $u(z).Q$ of P by $u(w).Q\{w/z\}$ or the replacement of a subterm $(\nu z)Q$ of P by $(\nu w)Q\{w/z\}$ where in each case w does not occur (at all) in Q.*
2. *Two processes P and Q are* α-convertible*, written $P =_\alpha Q$, if Q can be obtained from P by a finite number of changes of bound names.*

2.2. Late Labelled Transition Semantics

We propose herein a late labelled transition semantics of the π-calculus with polyadic synchronisation. In addition, we provide some examples that reflect the differences between this and the π-calculus.

Definition 2.4 *Late labelled transition relation*
Let $u = x_1 \cdot ... \cdot x_k$, where $k \in \mathbb{N}$. The late labelled transition relation $\xrightarrow{\alpha} \subseteq \mathcal{P}_S \times \mathcal{P}_S$*, where α is a possible action, is the smallest relation generated by the set of rules in Figure 2 (page 26).*[4]

The rules follow in a straightforward manner those of the π-calculus where we now consider vectors of names as channels. The rules follow in a straightforward manner those of the π-calculus, considering now vectors of names as channels. Note once again that the restriction rule, RES, considers singular and not composite names, *i.e.*, restriction is *partial*. Nevertheless, we enforce an all-or-nothing behavior, that is, we require the match of all the names in the vector channel to allow synchronization. The following example reflects the consequences of this type of restriction.

Example 2.5 *Let $P = (\nu x_1)\overline{x_1 \cdot x_2}\langle y\rangle$ and $Q = \overline{x_1 \cdot x_2}\langle y\rangle$. Then, P cannot perform the input action because of the restriction in one of its channel names, while Q can. Consider now $P = x(y)\overline{y \cdot z}\langle v\rangle \mid \overline{x}\langle w\rangle$. Its reduction performs a substitution in (only) one of the channel names, yielding $\overline{w \cdot z}\langle v\rangle$.*

Now that we have introduced the labelled transition relation for processes in the π-calculus with polyadic synchronisation, we can reflect on the importance of α-convertibility. The following example accounts for the relevance of this operation.

Example 2.6 *Let $P = (S \mid Q) \mid R$ where $S = x \cdot z(y).y(b)$, $Q = z \cdot y(b)$ and $R = \overline{x \cdot z}\langle c\rangle$. We expect S to synchronise with R in such a way that $P \xrightarrow{\tau} (c(b) \mid z \cdot w(b)) \mid \mathbf{0}$, but in order for this to be achievable we need to perform an α-conversion, else the side condition of the PAR1 rule is not satisfied.*

$$\dfrac{\dfrac{\dfrac{\dfrac{--}{x \cdot z(a).a(b) \xrightarrow{x \cdot z(a)} a(b)}\,\text{PREFIX}}{x \cdot z(y).y(b) \xrightarrow{x \cdot z(a)} a(b)}\,\text{CONV}}{x \cdot z(y).y(b) \mid z \cdot y(b) \xrightarrow{x \cdot z(a)} a(b) \mid z \cdot y(b)}\,\text{PAR1} \qquad \dfrac{--}{\overline{x \cdot z}\langle c\rangle \xrightarrow{\overline{x \cdot z}\langle c\rangle} \mathbf{0}}\,\text{PREFIX}}{P \xrightarrow{\tau} (c(b)) \mid z \cdot w(b)) \mid \mathbf{0}}\,\text{COMM}$$

Note that if we had not performed the α-conversion and had disrespected the side condition of the PAR1 rule then P would have evolved through a τ action into $c(b) \mid z \cdot c(b)$.

[4]Note that not included in the figure are four rules: the symmetric form CH2 of CH1 which has $Q + P$ instead of $P + Q$, and the symmetric forms PAR2, COMM2 and CLOSE2 of PAR1, COMM1, CLOSE1 in which the roles of the left and right components are swapped.

3. OBSERVATIONAL SEMANTICS

In this section we develop the behavioural theory of epi. In short, we define a contextual equivalence for epi—barbed congruence—and find its co-inductive characterisation. Following the literature of the π-calculus [MPW92, San96], with the necessary adjustments we introduce the "standard" notions of bisimilarity: ground, late, early, and open. Then study their preservation by the operators of epi and interrelate these notions, getting a lattice of discriminating power. Finally, we show that early congruence (early bisimilarity closed for all substitutions) coincides with barbed congruence, being thus its co-inductive characterisation.

Although not surprising, these results are technically difficult, and some proofs deviate from those in the π-calculus. This work is necessary to provide to epi behavioural theory. Appendix A contains the more complex proofs of the results presented in this section.

3.1. A Contextual Equivalence

Definitions. We define first an equivalence notion.

Definition 3.1 *Barbs*
The predicate 'P exhibits barb *β', written $P \downarrow_\beta$, is defined by:*

- *$P \downarrow_u$ if P can perform an input action on channel u*
- *$P \downarrow_{\overline{u}}$ if P can perform an output action on channel u*

A *barb* is an input or output channel identifier. Note that the predicate just defined concerns only visible and immediate possible action.

Example 3.2 *Let $P = (x \cdot y(a) \mid \overline{x \cdot y}\langle b\rangle\,).\overline{z}\langle c\rangle \;+\; \overline{w \cdot x}\langle c\rangle$ and $Q = (\nu x)(x \cdot y(a) \mid \overline{x \cdot y}\langle b\rangle\,).\overline{z}\langle c\rangle$, where all names are distinct. Then, $P \downarrow_{x\cdot y}$, $P \downarrow_{\overline{x\cdot y}}$, and $P \downarrow_{\overline{w\cdot x}}$, but $P \not\downarrow_{\overline{z}}$. Furthermore, Q exhibits no barbs.*

We now introduce the notion of barbed bisimilarity as proposed by Milner and Sangiorgi [MS92]. It is an equivalence relation on processes based on their observable behaviour.

Definition 3.3 *Barbed bisimilarity*

1. *A binary symmetric relation $\mathcal{S}$ is a* barbed bisimulation *if $P\mathcal{S}Q$ implies:*
   - *if $P \downarrow_\beta$ then $Q \downarrow_\beta$ for each barb β*
   - *if $P \xrightarrow{\tau} P'$ then there exists a Q' such that $Q \xrightarrow{\tau} Q'$ and $P'\mathcal{S}Q'$*
2. *Processes P and Q are* barbed bisimilar *if $P\mathcal{S}Q$ for some barbed bisimulation $\mathcal{S}$.*
3. Barbed bisimilarity*, written $\sim_b$, is the greatest barbed bisimulation.*

Properties. Note that barbed bisimilarity is not a congruence since it is not preserved by parallel composition, nor by replication, nor by substitution, as seen in the following examples.[5]

Example 3.4 *Let $P = \overline{m}\langle n\rangle\,.\overline{m}\langle n\rangle$, $Q = \overline{m}\langle n\rangle$ and $R = m(x)$. As seen in the previous example $P \sim_b Q$, and trivially $R \sim_b R$. Nonetheless, $P \,|\, R \not\sim_b Q \,|\, R$ since $P \,|\, R \xrightarrow{\tau} P' = \overline{m}\langle n\rangle$ and $P' \downarrow_{\overline{m}}$ but $Q \,|\, R \xrightarrow{\tau} \mathbf{0}$.*

Example 3.5 *Let $P = \overline{m}\langle n\rangle\,.\overline{a}\langle b\rangle \;+ m(x)$, $Q = \overline{m}\langle n\rangle\,.\overline{b}\langle a\rangle \;+ m(x)$ and a, b be distinct names. Then, P and Q are barbed bisimilar since they exhibit exactly the same barbs: $\overline{m}$ and m. Nonetheless, $!P$ and $!Q$ are not barbed bisimilar since two copies of P and two copies of Q can synchronise, but the resulting processes do not exhibit the same barb, i.e., $!P \xrightarrow{\tau} P'$ and $P' \downarrow_{\overline{a}}$ but $!Q \xrightarrow{\tau} Q'$ and $Q' \downarrow_{\overline{b}}$.*

Example 3.6 *Let $P = \overline{m} \mid n$ and $Q = \overline{m}.n + n.\overline{m}$. Then, P and Q are barbed bisimilar since they have the same barbs. We only analyse the case when P starts: if $P \downarrow_{\overline{m}}$ then $Q \downarrow_{\overline{m}}$ and if $P \downarrow_n$ then $Q \downarrow_n$.*

[5] Example 3.6 is an exercise proposed in reference [SW01].

However, if we consider the substitution $\sigma = \{n/m\}$, we have that $P\sigma$ and $Q\sigma$ are not barbed bisimilar, since $P\sigma \xrightarrow{\tau}$ but $Q\sigma \not\xrightarrow{\tau}$.

Nonetheless, barbed bisimilarity is preserved by the remaining operators.

Proposition 3.7 *The relation $\sim_b$ is preserved by prefixing, restriction and choice operators.*

PROOF: In Appendix A.

□

Closing barbed bisimilarity for parallel composition yields an equivalence notion.

Definition 3.8 *Barbed equivalence*
Two processes P and Q are barbed equivalent*, written $\sim_{beq}$, if $P \mid R \sim_b Q \mid R$ for every process R.*

In order to define barbed congruence we must first introduce the notion of context. Contexts are processes with a "hole".

Definition 3.9 *Barbed congruence*

1. *A* context *is obtained when a 'hole' $[\cdot]$ replaces a process $P \in \mathcal{P}_S$.*
2. *The process obtained by replacing $[\cdot]$ in C by $P \in \mathcal{P}_S$, where C is a context, is denoted by $C[P]$.*
3. *Two processes P and Q are* barbed congruent*, written $\simeq_b$, if $C[P] \sim_b C[Q]$ for every context $C[\cdot]$.*

We now extend to epi the result that establishes an alternative definition of barbed congruence in the π-calculus, as done by Sangiorgi and Walker [SW01]: closing barbed equivalence for substitution yields barbed congruence.

Lemma 3.10 *$P \simeq_b Q$ if and only if $P\sigma \sim_{beq} Q\sigma$ for any substitution σ*

PROOF: In Appendix A.

□

3.2. Four Notions of Bisimilarity

Seeking for a co-inductive characterisation of barbed congruence, we define the usual notions of bisimilarity, and inter-relate them.

Ground Bisimilarity. The first notion we will consider is that of ground bisimilarity, where there is no name instantiation.

Definition 3.11 *Ground bisimilarity*

1. *A binary symmetric relation $\mathcal{S}$ is a* ground bisimulation *if $P\mathcal{S}Q$ implies:*
   if $P \xrightarrow{\alpha} P'$ where $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P, Q) = \emptyset$ then there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.
2. *Processes P and Q are* ground bisimilar *if $P\mathcal{S}Q$ for some ground bisimulation $\mathcal{S}$.*
3. Ground bisimilarity*, written $\sim_g$, is the largest ground bisimulation.*[6]

The notion of ground bisimilarity is very simple since a process merely has to imitate the other in its possible transitions and vice versa without considering name instantiation. Unfortunately, as in the π-calculus, a consequence of this is that ground bisimilarity is not preserved by the parallel composition operator, as the following example shows.

[6] The existence and uniqueness of a largest bisimulation is a direct consequence of the Knäster-Tarski's Fixed Point Theorem.

Example 3.12 *Let* $P = (\nu a)(z(w).\overline{a \cdot w}\langle c\rangle \mid a \cdot y(b))$ *and* $Q = z(w)$*. Then both* P *and* Q *are ground bisimilar since after performing the input action they both become inactive. Conversely,*

$$P' = P \mid \overline{z}\langle y\rangle \xrightarrow{\tau} (\nu a)(\overline{a \cdot y}\langle c\rangle \mid a \cdot y(b)),$$

which can also perform an internal action, while $Q' = Q \mid \overline{z}\langle y\rangle$ *can only perform one internal action and then becomes inactive. We can then conclude that although* P *and* Q *are ground bisimilar,* P' *and* Q' *are not ground bisimilar.*

Ground bisimilarity is not preserved by replication either, as the following example illustrates. Note that in the definition of P we use polyadic CCS-like prefixes $\overline{a \cdot w}$ and $a \cdot y$ where no item is being sent or expected to be received. We do this to highlight the fact that what could be transmitted is irrelevant, the problem lies in the synchronisation of the composite channels. In general, $\overline{u}.P$ will be used as shorthand for $\overline{u}\langle y\rangle .P$ for some y, and $u.P$ will be used as shorthand for $u(y).P$ where $y \notin \mathsf{fn}(P)$.

Example 3.13 *Let* $P = (\nu a)(z(w).\overline{a \cdot w}|a \cdot y).\overline{z}\langle x\rangle + \overline{z}\langle y\rangle$ *and* $Q = z(w) + \overline{z}\langle y\rangle$*, where* w *and* y *are distinct. Then* $P \sim_g Q$*, but* $!P \not\sim_g !Q$ *since two copies of* P *and two copies of* Q *can synchronise and the resulting processes are not bisimilar. In detail,* $!P \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\overline{z}\langle x\rangle} P'$*, but no descendant of* $!Q$ *can ever perform an output action* $\overline{z}\langle x\rangle$*.*

Nonetheless, ground bisimilarity is preserved, just like in the π-calculus, by the remaining operators.

Lemma 3.14 *The relation* $\sim_g$ *is preserved by prefixing, restriction and choice operators.*

PROOF: In Appendix A.

□

Late and Early Bisimilarity. We now introduce the notions of *late* and *early bisimilarity*, which differ in their treatment of name instantiation for input actions. The definitions of these notions are standard. In *late* bisimilarity we require that the derivative of a process simulates the derivative of the other process (and vice versa) for all possible instantiations of the bound parameter. It is called late because the choice of the name instantiation is made *after* the choice of the derivative.

Definition 3.15 *Late bisimilarity*
Let $u = x_1 \cdot ... \cdot x_k$ *where* $k \in \mathbb{N}$*.*

1. *A binary symmetric relation* $\mathcal{S}$ *is a* late bisimulation *if* $P\mathcal{S}Q$ *implies:*
   - *if* $P \xrightarrow{\alpha} P'$ *where* $\alpha = \overline{u}\langle y\rangle, \overline{u}(y)$ *or* τ *and* $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P,Q) = \emptyset$ *then there is a* Q' *such that* $Q \xrightarrow{\alpha} Q'$ *and* $P'\mathcal{S}Q'$*.*
   - *if* $P \xrightarrow{u(y)} P'$ *where* $y \notin \mathsf{fn}(P,Q)$ *then there is a* Q' *such that* $Q \xrightarrow{u(y)} Q'$ *and for each* w*,* $P'\{w/y\}\mathcal{S}Q'\{w/y\}$*.*
2. *Two processes* P *and* Q *are* late bisimilar *if* $P\mathcal{S}Q$ *for some late bisimulation* $\mathcal{S}$*.*
3. Late bisimilarity*, written* $\sim_l$*, is the largest late bisimulation.*

In *early* bisimilarity we require that under the same possible name instantiation there is a derivative of each of the processes that simulates the other and vice versa. It is named early because the choice of the name instantiation is made *before* the choice of the derivative.

Definition 3.16 *Early bisimilarity*
Let $u = x_1 \cdot ... \cdot x_k$ *where* $k \in \mathbb{N}$*.*

1. *A binary symmetric relation* $\mathcal{S}$ *is an* early bisimulation *if* $P\mathcal{S}Q$ *implies:*
   - *if* $P \xrightarrow{\alpha} P'$ *where* $\alpha = \overline{u}\langle y\rangle, \overline{u}(y)$ *or* τ *and* $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P,Q) = \emptyset$ *then there is a* Q' *such that* $Q \xrightarrow{\alpha} Q'$ *and* $P'\mathcal{S}Q'$*.*

- *if* $P \xrightarrow{u(y)} P'$ *where* $y \notin \mathsf{fn}(P,Q)$ *then for each* w *there is a* Q' *such that* $Q \xrightarrow{u(y)} Q'$ *and* $P'\{w/y\}\mathcal{S}Q'\{w/y\}$.

2. *Two processes* P *and* Q *are* early bisimilar *if* $P\mathcal{S}Q$ *for some early bisimulation* $\mathcal{S}$.
3. Early bisimilarity*, written* $\sim_e$*, is the largest early bisimulation.*

Similarly to what happens in the π-calculus, in the π-calculus with polyadic synchronisation both late and early bisimilarity are not preserved by input prefixing. This is evidenced by the following example where we consider processes in the π-calculus with biadic synchronisation.

Example 3.17 *Let* $P = (\nu a)(\overline{a \cdot z}\langle c\rangle \,|a \cdot y(b))$ *and* $Q = \mathbf{0}$. *Since both* P *and* Q *are unable to perform any action, we have that* P *and* Q *are late and early bisimilar. Now consider the processes* $P' = z(y).P$ *and* $Q' = z(y).\mathbf{0}$. *Then* $P' \xrightarrow{z(y)} P$ *and* $Q' \xrightarrow{z(y)} \mathbf{0}$, *but we have that* $P\{z/y\} \xrightarrow{\tau}$ *while* $\mathbf{0} \overset{\tau}{\not\longrightarrow}$. *Thus,* P' *and* Q' *are neither early nor late bisimilar.*

Again, as in the π-calculus, both late and early bisimilarity are preserved by all other operators.

Proposition 3.18 *The relations* $\sim_l$ *and* $\sim_e$ *are preserved by all operators except input prefixing.*

PROOF: The proof follows that for the π-calculus by Milner *et al.* for the output prefixing, choice and parallel composition operators [MPW92] and that by Milner for the replication operator [Mil93]. Example 3.17 proves that neither $\sim_l$ nor $\sim_e$ are preserved by input prefixing.

□

Moreover, as in the original π-calculus, congruences for late and early bisimilarity, $\simeq_l$ and $\simeq_e$, are achieved by closing the equivalences over all name substitutions [MPW92]. The relation between the notions of late bisimilarity and late congruence, and of early bisimilarity and early congruence, are shown in the following proposition.

Proposition 3.19 $\simeq_l \subset \sim_l$ *and* $\simeq_e \subset \sim_e$.

PROOF: The inclusion follows directly from the definitions of late bisimilarity and late congruence because for $P, Q \in \mathcal{P}_S$, if $P \simeq_l Q$ then for all substitutions σ we have that $P\sigma \sim_l Q\sigma$. In particular, this is true for the identity substitution, that is, $P \sim_l Q$.

The following example is evidence of the strictness of the inclusion. Let $P, Q \in \mathcal{P}_S$ and consider distinct $x, y, z, w \in N$. If $P = (\nu w)(\overline{w \cdot x}\langle a\rangle \,|w \cdot y(b))$ and $Q = (\nu w)(\overline{w \cdot x}\langle a\rangle \,|w \cdot z(b))$, then $P \sim_l Q$ since both processes are inactive. Nonetheless, for $\sigma = \{y/x\}$, $P\sigma$ can perform a τ action, but $Q\sigma$ remains inactive. Thus, $P\sigma \not\sim_l Q\sigma$, and so $P \not\simeq_l Q$.

□

Proposition 3.20 $\simeq_e \subset \sim_e$

PROOF: The inclusion follows directly from the definitions of early bisimilarity and early congruence and is similar to that of Proposition 3.19. The same example given in Proposition 3.19 can be used to prove the strictness of the inclusion, since $P \sim_e Q$ but $P \not\simeq_e Q$.

□

Open Bisimilarity. The notion of open bisimilarity was introduced by Sangiorgi and proved to be a congruence relation in the π-calculus [San96]. That is also the case here: in epi, open bisimilarity is a congruence.

Definition 3.21 *Open bisimilarity*

1. *A binary symmetric relation $\mathcal{S}$ is an* open bisimulation *if $P\mathcal{S}Q$ implies for every substitution σ: If $P\sigma \xrightarrow{\alpha} P'$ where $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P\sigma, Q\sigma) = \emptyset$ then there is a Q' such that $Q\sigma \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.*
2. *Two processes P and Q are* open bisimilar *if $P\mathcal{S}Q$ for some open bisimulation $\mathcal{S}$.*
3. Open bisimilarity*, written $\sim_o$, is the largest open bisimulation.*

As expected, open bisimilarity is a congruence.

Proposition 3.22 *The relation $\sim_o$ is preserved by all operators.*

PROOF: The proof follows that in [San96].

□

3.3. Expressiveness Results

The congruence properties appear to stem directly from those of the π-calculus. However, ground bisimilarity is a full congruence in the asynchronous π-calculus without match [San00] (and a similar result holds for late and for early bisimilarity [HHK95]), but this result does *not* hold if we consider the asynchronous π-calculus with polyadic synchronisation, as seen in Example 3.12. Matching does not need to be considered as a primitive in the π-calculus with polyadic synchronisation (synchronous or asynchronous) since it can be derived. Therefore, ground, late and early bisimilarities are *not* congruences in the asynchronous π-calculus with polyadic synchronisation (without match).

We now analyse the relationships between the bisimilarity relations previously defined and present a general diagram that summarises these results in Corollary 3.27. The results and proofs are similar to those presented for the π-calculus [MPW92, Qua99]. The largest open bisimulation is itself a late bisimulation, and it is also included in late congruence.

Proposition 3.23 *$\sim_o \subset \sim_l$ and $\sim_o \subset \simeq_l$.*

Late bisimilarity is itself an early bisimulation, although the reverse does not hold. The same result holds if we consider the notions of late and early congruences instead of late and early bisimilarity.

Proposition 3.24 *$\sim_l \subset \sim_e$ and $\simeq_l \subset \simeq_e$.*

Our last result shows that if two processes are early bisimilar then they are also ground bisimilar, although the reverse does not hold.

Proposition 3.25 *$\sim_e \subset \sim_g$.*

Barbed bisimilarity is a much coarser relation than the bisimilarities introduced so far. The following example illustrates the difference between barbed bisimilarity and those notions of bisimilarity.

Example 3.26 *Let $P = \overline{m}\langle n\rangle.\overline{m}\langle n\rangle$ and $Q = \overline{m}\langle n\rangle$. Then, P and Q are barbed bisimilar since their only barb is $\overline{m}$. However, P and Q are not ground, nor late, nor early nor open bisimilar since $P \xrightarrow{\overline{m}\langle n\rangle} \overline{m}\langle n\rangle$ and $Q \xrightarrow{\overline{m}\langle n\rangle} \mathbf{0}$, which are obviously not bisimilar.*

We now summarise the results presented in the following diagram where $\rightarrow$ stands for strict inclusion $\subset$.

Corollary 3.27

$$\begin{array}{ccccccc} \sim_o & \rightarrow & \sim_l & \rightarrow & \sim_e & \rightarrow & \sim_g \\ & \searrow & \uparrow & & \uparrow & \nearrow & \\ & & \simeq_l & \rightarrow & \simeq_e & & \end{array}$$

A co-inductive characterisation of barbed congruence. We finally prove that the "natural" contextual equivalence—barbed congruence—coincides with early bisimilarity, which is thus a co-inductive characterisation of the latter. Sangiorgi obtained a characterisation of barbed equivalence by proving it coincided with early bisimilarity [San92]. We extend that result for the π-calculus with polyadic synchronisation, completing the behavioural theory.

Theorem 3.28 $\sim_e = \sim_{beq}$

PROOF: In Appendix A.

□

Corollary 3.29 $\simeq_e = \simeq_b$

PROOF: Let $P, Q \in \mathcal{P}_S$.

($\Rightarrow$) If $P \simeq_e Q$ then by definition of early congruence, $P\sigma \sim_e Q\sigma$ for any substitution σ. By Theorem 3.28 we know that $P\sigma \sim_{beq} Q\sigma$, and by Lemma 3.10 $P \simeq_b Q$.

($\Leftarrow$) If $P \simeq_b Q$ then by Lemma 3.10 we know that $P\sigma \sim_{beq} Q\sigma$ for any substitution σ. By Theorem 3.28 we have that $P\sigma \sim_e Q\sigma$, and therefore $P \simeq_e Q$.

□

4. ENCODING CRYPTOGRAPHIC PRIMITIVES

To model an encryption system one needs to consider primitives to encrypt and decrypt information using keys. In symmetric cryptography these primitives should respect the following equation, where m is the information to be encoded and k is the shared key.

$$\mathsf{decrypt}(\mathsf{encrypt}(m, k), k) = m \tag{1}$$

In asymmetric cryptography, one needs an extra primitive (*e.g.*, pubk) to obtain a public key from a private one. The equation defining the intended system' behaviour, considering now k to be a private key, is as follows.

$$\mathsf{decrypt}(\mathsf{encrypt}(m, \mathsf{pubk}(k)), k) = m \tag{2}$$

4.1. Cryptographic Primitives in Process Calculi

To our knowledge, the first mention of a possible encoding of a calculus with (symmetric) cryptographic primitives into a calculus with polyadic synchronisation was put forth by Abadi and Gordon [AG97]. The idea can be summarised in the following way: the sending of a message m encrypted under a key k over a channel a can be seen as $\overline{a \cdot k}\langle m \rangle .P$. In order to receive this message, the other party needs to know the channel where the message is being transmitted and the key, which could be represented as $a \cdot k(m).P$.

An encoding of key encryption primitives into π-calculus with polyadic synchronisation is proposed by Carbone and Maffeis in the introduction of their paper to further illustrate its expressive power [CM03].

$$[\!|\, \mathsf{encrypt}\, m \hookrightarrow^k x \,\mathsf{in}\, P \,|\!] \stackrel{\text{def}}{=} (\nu x)(!\overline{x \cdot k}\langle m \rangle \mid [\!|\, P \,|\!])$$

$$[\!|\, \mathsf{decrypt}\, x \hookrightarrow^k m \,\mathsf{in}\, P \,|\!] \stackrel{\text{def}}{=} x \cdot k(m).[\!|\, P \,|\!]$$

With this definitions, the symmetric cryptography law (Equation 1 above), translated to this new language, is very easy to prove: the decryption of the encrypted text m (with the shared key k) should be equal to m. In that case, m is made available (as an input process).[7]

[7] Recall that the matching construct is encodable in epi.

Proposition 4.1

$$[|\, \mathsf{encrypt}\, m \looparrowright^{k} x \,\mathsf{in}\, (\mathsf{decrypt}\, x \looparrowright^{k} y \,\mathsf{in}\, (\mathsf{if}\ m = y\ \mathsf{then}\ m().\mathbf{0}\,))\,|]\ \simeq_e\ m().\mathbf{0}$$

PROOF: In Appendix C.

□

This modelling mechanism is not restricted to symmetric cryptography: if one provides a way to recover a public key from a private (or secret) one (hence implementing the primitive pubk), these primitives also support asymmetric cryptography. Following the Applied π-calculus approach, assume that every principal of a security protocol is a process providing a private channel (*e.g.*, $pfsk$, meaning public from secret key) which, given a channel representing a private key and a reply channel, test if the private key is the correct one and if so, sends the public key on the reply channel.

Consider thus the following extra primitive 'public on z from private x', which makes available a(n input) channel ($pfsk$) that sends on a received reply channel (z) the public key (pk) corresponding to the private key (sk), if the value received (x) is indeed the private key. This primitive is easily encodable in epi, parameterising the map with a function associating to a channel a pair of keys.

$$[|\, \mathsf{public\ on}\, z\, \mathsf{from\ private}\, x\, \mathsf{in}\, P\,|]_{\{pfsk \mapsto (pk,sk)\}} \stackrel{\text{def}}{=} !pfsk(x,z).\mathsf{if}\ x = sk\ \mathsf{then}\ (\overline{z}\langle pk\rangle\ |\ [|\, P\,|]_{\{(pk,sk)\}})$$

The asymmetric cryptography law (Equation 2 above), translated to this setting, holds as well. Encryption uses the inverse function, recovering the private key from the public one.

$$[|\, \mathsf{private\ on}\, k\, \mathsf{from\ public}\, x\,|]_{\{(pk,sk)\}} \stackrel{\text{def}}{=} !\,\mathsf{if}\ k = pk\ \mathsf{then}\ x \cdot pk(y).\overline{x \cdot sk}\langle y\rangle$$

Proposition 4.2

$$\begin{aligned}(\nu pfsk, sk, z)(&\overline{pfsk}\langle sk, z\rangle\ |\\ &[|\, \mathsf{public\ on}\, z\, \mathsf{from\ private}\, x\ \mathsf{in}\\ &\quad z(k).\mathsf{encrypt}\, m \looparrowright^{k} x\ \mathsf{in}\ (\quad \mathsf{private\ on}\, k\, \mathsf{from\ public}\, x\ |\\ &\qquad\qquad \mathsf{decrypt}\, x \looparrowright^{sk} y\ \mathsf{in}\ (\mathsf{if}\ m = y\ \mathsf{then}\ m().\mathbf{0}\,))\,|]_{\{pfsk \mapsto (pk,sk)\}})\\ \simeq_e\ & m().\mathbf{0}\end{aligned}$$

PROOF: In Appendix C.

□

4.2. Cryptographic epi

Carbone and Maffeis did not define the semantics of the primitives, and thus did not study the properties of the encoding (as moreover, they have not developed the behavioural theory of epi). Herein we do all that work: we first add to epi two key encryption primitives, encrypt and decrypt, defining the cryptographic π-calculus with polyadic synchronisation (crypto-epi), and extend epi labelled transition system with rules dealing with these new constructs. Then we show that the new constructors preserve the bisimilarity relations defined to epi, and finally, we prove that these cryptographic primitives are derivable constructs: crypto-epi can be fully abstractly encoded in epi; thus we prove that the original calculus does not need to be extended with those primitives, at least from the point of view of expressiveness. Moreover, since the encoding is fully abstract, crypto-epi enjoys of all the behavioural theory of epi. The main achievement here is thus a mobile calculus with cryptographic primitives enjoying the "standard" behavioural theory. Adapting analysis tools like the Mobility Workbench [Vic94, VM94] should be straightforward.

We start by extending the language we have been working with, adding cryptographic primitives.

Syntax. Consider two extra productions in the syntax of epi (*cf.* Definition 2.1): $\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, P$ and $\mathsf{decrypt}\, x \twoheadrightarrow^k m \,\mathsf{in}\, P$. The first construct non-deterministically encrypts the cipher text m under key k and returns the encrypted message as the fresh name x, to be used in the scope of P, where it occurs bound. The decryption of message x through the key k binds the name m in the continuation P to the original message. Notice that one, when encrypting, does not expect free occurrences of m and k in P; and when decrypting, does not expect free occurrences of x and k in P. As shown above, these primitives support both symmetrical and asymmetrical encryption systems.

Labelled Transition Semantics. The rules of epi in Figure 2 (page 26), together with the rules in Figure 3 inductively define the transition semantics of crypto-epi.

The behavioural theory of epi extends naturally to this new setting. The notions of bisimilarity, introduced in the previous section, enjoy similar properties when consider the new constructs. Notice that the decrypt primitive behaves like an input prefix, thus it does not preserve ground, early or late bisimilarity, but naturally, it preserves open bisimilarity. The notion of early congruence in crypto-epi is obtained in the same manner, and the results in Corollary 3.29 also extend straightforwardly to this new setting. Therefore, the theory developed can be used to analyse and (equationally) prove properties of security protocols.

Equational laws. To present some examples we need to introduce some results. First, notice that syntactical equality is an early bisimilarity, and that any strong bisimilarity is strictly included in the corresponding weak version. In particular, $=\subset\simeq_e\subset\approx_e$.

Second, the usual structural congruence laws of the π-calculus [MPW92, SW01] also hold in any bisimilarity. Therefore, we use below instances of the following laws.[8]

Lemma 4.3 (Structural Laws)

1. $(\mathcal{P}_S, |, \mathbf{0})$ *is a commutative monoïd with respect to* $\simeq_e$.
2. $(\nu x)\mathbf{0} \simeq_e \mathbf{0}$ *and* $(\nu x)!\overline{x \cdot k}\langle m\rangle \simeq_e \mathbf{0}$
3. $(\nu x)(P \mid Q) \simeq_e (P \mid (\nu x)Q)$, *if* $x \notin \mathsf{fn}(P)$

Finally, the following laws are useful in our setting.

Lemma 4.4

1. $\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, P \simeq_e (\nu x)(!\overline{x \cdot k}\langle m\rangle \mid P)$
2. $\mathsf{decrypt}\, x \twoheadrightarrow^k m \,\mathsf{in}\, P \simeq_e x \cdot k(m).P$

PROOF: Construct the respective bisimulations containing the pair in question and, in the two last cases, the identity relation on processes.

□

4.3. A secure message exchange

Sending a value in a free (*i.e.* public) channel is *insecure*, as any context (*i.e.* observer) can have access to it. Bound (*i.e.* private) channels are, in this framework, consider secure. Since one often needs to send sensitive data in public channels, we would like to show two basic properties: (1) decrypting an encrypted value with the correct key gives back the original value, and no other key produces it; and (2) sending encrypted values in public channels is secure, as observers without the right keys cannot decrypt them.

To illustrate the use of these properties (and their correctness), consider a cryptographic protocol for secure message exchange, proposed by Carbone and Maffeis [CM03], defined as $(\nu sec)(P \mid Q)$ where P and Q

[8] Instead of proving each of these laws one may prove the "Harmony Lemma", allowing to establish that structural congruence is a bisimulation.

are the following processes.

$$P \stackrel{\text{def}}{=} (\nu k)\overline{sec}\langle k\rangle\,.public(y).\mathsf{decrypt}\, y \looparrowright^{k} w \,\mathsf{in}\, R$$

$$Q \stackrel{\text{def}}{=} (\nu m)sec(z).\mathsf{encrypt}\, m \looparrowright^{z} x \,\mathsf{in}\, \overline{public}\langle x\rangle\,.S$$

Assume that sec does not occur free neither in R nor in S, m does not occur free in R, and k and z do not occur free in S.

We show the correctness of the protocol (with respect to weak bisimilarity, to ignore silent moves): an external observer cannot get neither the key k nor the clear text message m during the execution of the protocol since the transfer of the knowledge of the key is done on a secure—since private—channel (sec). Moreover, decrypting the encrypted value x with the key k (and with it only) gives back the original value m.

The following equation captures the correctness of the protocol.

$$(\nu sec)(P \mid Q) \approx_e (\nu k, m)(R\{m/w\} \mid \mathsf{encrypt}\, m \looparrowright^{k} x \,\mathsf{in}\, S)$$

The analysis below proves the equation. Note that the protocol is deterministic.

1. Consider the following processes.

$$P' \stackrel{\text{def}}{=} public(y).\mathsf{decrypt}\, y \looparrowright^{k} w \,\mathsf{in}\, R$$

$$Q' \stackrel{\text{def}}{=} (\nu m)\mathsf{encrypt}\, m \looparrowright^{k} x \,\mathsf{in}\, \overline{public}\langle x\rangle\,.S\{k/z\}$$

   The first step is the transmission of the key on channel sec:

$$(\nu sec)(P \mid Q) \xrightarrow{\tau} (\nu sec, k)(P' \mid Q')$$

2. Consider now the following processes.

$$P'' \stackrel{\text{def}}{=} \mathsf{decrypt}\, x \looparrowright^{k} w \,\mathsf{in}\, R\{x/y\}$$

$$Q'' \stackrel{\text{def}}{=} (\nu m)(!\overline{x \cdot k}\langle m\rangle \mid S\{k/z\})$$

   The next step is the transmission of the encrypted message:

$$(\nu sec, k)(P' \mid Q') \xrightarrow{\tau} (\nu sec, k, x)(P'' \mid Q'')$$

3. Finally, the encrypted message is decrypted:

$$(\nu sec, k, x)(P'' \mid Q'') \xrightarrow{\tau} (\nu sec, k, x, m)(R\{x/y\}\{m/w\} \mid (!\overline{x \cdot k}\langle m\rangle \mid S\{k/z\}))$$

Since $x \notin \mathsf{fn}(R)$ and $k, m \notin \mathsf{fn}(S)$, then $R\{x/y\} = R$ and $S\{k/z\} = S$. Moreover, $sec \notin \mathsf{fn}(R) \cup \mathsf{fn}(S)$. Thus, using the laws presented above, one concludes the proof by transitivity.

$$\begin{aligned}
&(\nu sec, k, x, m)(R\{x/y\}\{m/w\} \mid (!\overline{x \cdot k}\langle m\rangle \mid S\{k/z\})) \\
&= (\nu sec, k, x, m)(R\{m/w\} \mid (!\overline{x \cdot k}\langle m\rangle \mid S)) \\
&\simeq_e (\nu k, m)(R\{m/w\} \mid (\nu x)(!\overline{x \cdot k}\langle m\rangle \mid S)) \\
&\simeq_e (\nu k, m)(R\{m/w\} \mid \mathsf{encrypt}\, m \looparrowright^{k} x \,\mathsf{in}\, S)
\end{aligned}$$

5. A FULLY ABSTRACT ENCODING

In order to prove the soundness and completeness of the encoding with respect to barbed congruence, which we proved in Corollary 3.29 to coincide with early congruence, we build on successive auxiliary results. More elaborate proofs are in Appendix B.

Henceforth, whenever we write P we refer to a process of the cryptographic π-calculus with polyadic synchronisation.

5.1. Operational correspondence

To ease the proofs ahead we use the notions of *structural congruence* and of *bisimulation up to* (the latter introduced for CCS [Mil89]). Note that the first lemma concerns the commonly denoted *structural properties* which are preserved by the ground bisimulation.

Definition 5.1 *Structural congruence*

1. Structural congruence*, written* $\equiv$*, is the smallest congruence on the processes that satisfies the following axioms where* P*,* Q*,* $R \in \mathcal{P}_S$ *and* z*,* $w \in N$*.*
   - $P + (Q + R) \equiv (P + Q) + R$
   - $P + Q \equiv Q + P$
   - $P + \mathbf{0} \equiv P$
   - $P|(Q|R) \equiv (P|Q)|R$
   - $P|Q \equiv Q|P$
   - $P|\mathbf{0} \equiv P$
   - $(\nu z)\mathbf{0} \equiv \mathbf{0}$
   - $(\nu z)(\nu w)P \equiv (\nu w)(\nu z)P = (\nu w, z)P$
   - $(\nu z)(P|Q) \equiv P|(\nu z)Q$ *if* $z \notin \mathsf{fn}(P)$
2. *Any two processes related by these axioms are called* structurally congruent*.*

Lemma 5.2 [9] *Let* P*,* $Q \in \mathcal{P}_S$*. If* $P \equiv Q$ *then* $P \sim_g Q$*.*

Definition 5.3 [10]*Ground bisimulation up to* $\sim_g$
A binary symmetric relation $\mathcal{S}$ *is a* ground bisimulation up to $\sim_g$*, if* $P\mathcal{S}Q$ *implies:*

- *if* $P \xrightarrow{\alpha} P'$ *where* $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P, Q) = \emptyset$ *then there is a* Q' *such that* $Q \xrightarrow{\alpha} Q'$ *and* $P' \sim_g \mathcal{S} \sim_g Q'$*.*

Proposition 5.4 [11] *If* $P\mathcal{S}Q$ *where* $\mathcal{S}$ *is a ground bisimulation up to* $\sim_g$ *then* $P\mathcal{S}'Q$ *where* $\mathcal{S}'$ *is a ground bisimulation.*

PROOF: The proof follows that for CCS [Mil89] with the necessary adjustments since we are considering the π-calculus with polyadic synchronisation. The proof can be split into proving firstly that $\sim_g \mathcal{S} \sim_g$ is a ground bisimulation and secondly that $\mathcal{S}$ is included in $\sim_g \mathcal{S} \sim_g$.

□

The following lemma shows a strong operational correspondence between the actions of a process and the actions of its encoding.

Lemma 5.5 *Operational Correspondence*

1. *If* $[\![P]\!] \xrightarrow{\alpha} Q$ *then there is a* P' *such that* $P \xrightarrow{\alpha} P'$ *and* $[|P'|] = Q$*.*
2. *If* $P \xrightarrow{\alpha} P'$ *then* $[|P|] \xrightarrow{\alpha} [|P'|]$*.*

PROOF: In Appendix B.

□

5.2. Full Abstractness Result

The following lemmas prepares the ground for proving the soundness and the completeness of the encoding.

Lemma 5.6

[9] The same result holds for all the other notions of bisimilarity presented in this section.
[10] An analogous definition can be presented for all notions of bisimilarity introduced in this section.
[11] The same result holds for any of the notions of bisimilarity presented in this section.

1. *If* $[\![P]\!] \sim_e [\![Q]\!]$ *then* $P \sim_e Q$.
2. *If* $[\![P]\!] \sim_{beq} [\![Q]\!]$ *then* $P \sim_{beq} Q$.

PROOF: In Appendix B.

□

Lemma 5.7

1. *If* $P \sim_e Q$ *then* $[\![P]\!] \sim_e [\![Q]\!]$.
2. *If* $P \sim_{beq} Q$ *then* $[\![P]\!] \sim_{beq} [\![Q]\!]$.

PROOF: Similar to the one of the previous lemma.

□

We are now in a position to prove the main result of this paper: there is a fully abstract encoding of the cryptographic primitives in epi.

Theorem 5.8 *Soundness*
If $[\![P]\!] \simeq_b [\![Q]\!]$ *then* $P \simeq_b Q$

PROOF: If $[\![P]\!] \simeq_b [\![Q]\!]$ then for any substitution σ we have that $[\![P]\!]\sigma \sim_{beq} [\![Q]\!]\sigma$. By Lemma B.1 we then know that $[\![P\sigma]\!] \sim_{beq} [\![Q\sigma]\!]$, and by Lemma 5.6.2 we have that $P\sigma \sim_{beq} Q\sigma$.

□

Theorem 5.9 *Completeness*
If $P \simeq_b Q$ *then* $[\![P]\!] \simeq_b [\![Q]\!]$.

PROOF: If $P \simeq_b Q$ then for any substitution σ we have that $P\sigma \sim_{beq} Q\sigma$. By Lemma 5.7.2 then $[\![P\sigma]\!] \sim_{beq} [\![Q\sigma]\!]$ and by Lemma B.1 we know that $[\![P\sigma]\!] = [\![P]\!]\sigma$, thus $[\![P]\!]\sigma \sim_{beq} [\![Q]\!]\sigma$, *i.e.*, $[\![P]\!] \simeq_b [\![Q]\!]$.

□

6. CONCLUSIONS AND FUTURE WORK

The various variants of π-calculus possess a very rich behavioural theory, with contextual equivalences characterised by bisimulations, and with axiomatic laws for reasoning about programs. However, the extra structure for data handling in cryptographic calculi like the Applied π-calculus or Spi, severely complicates equational reasoning: naïve adaptation of bisimulations are not adequate; new notions developed are "heavy", and difficult to automate [AF01, AG97, BAF07, BNP02, BN05].

Our contribution is this: we provide standard behavioural theory for a mobile calculus with (non-deterministic, symmetrical or asymmetrical) key encryption primitives. We show the expressiveness of the primitives by proving both thesymmetrical and the asymmetrical encryption systems laws, and by proving the correctness of a small protocol.

This work may be used further not only to directly analyse security protocols (possibly defining other cryptographic primitives), but also to study the relationship with the other calculi, comparing the observational equivalences and trying to define encodings. Moreover, adapting analysis tools like the Mobility Workbench [Vic94, VM94] should be straightforward.

Aim and achievements. One aim of this work is to show that the π-calculus with polyadic synchronisation, epi, is expressive enough to provide behavioural theory for the study of cryptographic protocols. In particular, we show that, in epi, explicit encryption and decryption primitives (handy for

specifying protocols, but a burden when developing behavioural theory) are not needed because they may be fully abstractly encoded. Thus, they may be simply defined as programming constructs, what simplifies the development of the behavioural theory and of analysis tools.

To attain this aim, we study in detail the behavioural semantics of epi. We first define a contextual equivalence — barbed congruence — and look for a co-inductive congruence relation which characterises it. To obtain such a result, we define in epi the usual notions of bisimilarities proposed for the π-calculus, and comparing them, establishing a lattice of inter-relations (similar to that of the π-calculus). We establish that, in epi, barbed congruence, the "natural" contextual equivalence, coincides with early bisimilarity. Moreover, we extend epi with non-deterministic, symmetrical, cryptographic primitives, defining the syntax and operational semantics of this new calculus. The behavioural theory also extends naturally to this setting. Following Carbone and Maffeis [CM03] we define an encoding of the new constructs for encryption and decryption of messages into the original epi. Furthermore, we prove that such an encoding is sound and complete with respect to barbed congruence. This fully abstract encoding allows to import to crypto-epi all the behavioural theory of epi. We therefore conclude that the π-calculus with polyadic synchronisation (epi) is expressive enough to provide behavioural theory for, to analyse and to verify, security protocols. To illustrate the use of the theory developed, we prove the correctness of a protocol of secure message exchange. This work strengthens the hypothesis that a fully abstract encoding of a crypto calculus like the Spi-calculus into epi is possible. Notice that Baldamus *et al.* already proposed an encoding of Spi into the pi-calculus, but only preserving may testing [BPV04].

Future work. We plan to study if and how epi can express properties of cryptographic protocols such as authenticity and secrecy. In particular, we shall address the following issues:

1. adapt the Mobility Workbench to work with this setting;
2. develop equational (axiomatic) theory;
3. test with larger examples / known protocols;
4. deal with other crypto primitives; and
5. study an encoding of Spi and/or of Applied Pi into epi.

ACKNOWLEDGEMENTS

This research has initially been carried out in the context of Joana Martinho's Master in Software Systems Engineering at Aalborg University, supervised by Luca Aceto and António Ravara. Special thanks to Luca for all the support and guidance on this research.

We also thank Michele Boreale, Marco Carbone, Sergio Maffeis, and the anonymous referees for useful comments on some of the matters discussed herein.

António Ravara was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia and the EU IST proactive initiative FET-Global Computing (project Sensoria, IST–2005–16004).

REFERENCES

[[AF01]] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

[[AG97]] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[[BAF07]] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 2007. To appear.

[[BN05]] Johannes Borgström and Uwe Nestmann. On bisimulations for the spi calculus. *Mathematical Structures in Computer Science*, 15(3):487–552, 2005.

[[BNP02]] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. *SIAM Journal on Computing*, 31(3):947–986, 2002.

[[BPV04]] Michael Baldamus, Joachim Parrow, and Björn Victor. Spi calculus translated to π-calculus preserving may-testing. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS '04)*, pages 21–31. IEEE Computer Society Press, 2004.

[[CM03]] Marco Carbone and Sergio Maffeis. On the expressive power of polyadic synchronization in π-calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.

[[HHK95]] Martin Hansen, Hans Hüttel, and Josva Kleist. Bisimulations for asynchronous mobile processes. In *Proceedings of the Tiblisi Symposium on Language, Logic, and Computation*. Research paper HCRC/RP-72, Human Communication Research Centre, University of Edinburgh, 1995.

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

[[Mil93]] Robin Milner. The polyadic π-calculus: A tutorial. In Friedrich L. Bauer, Wilfried Brauer, and Helmut Schwichtenberg, editors, *Logic and Algebra of Specification, Proceedings of the International NATO Summer School (Marktoberdorf, Germany, 1991)*, volume 94 of *Series F: Computer and System Sciences*. NATO Advanced Study Institute, Springer-Verlag, 1993. Available as Technical Report ECS-LFCS-91-180, University of Edinburgh, U. K., 1991.

[[MPW92]] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Journal of Information and Computation*, 100:1–77, 1992. Available as Technical Reports ECS-LFCS-89-85 and ECS-LFCS-89-86, University of Edinburgh, U. K., 1989.

[[MS92]] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP '92)*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer-Verlag, 1992.

[[Qua99]] Paola Quaglia. The pi-calculus: Notes on labelled semantics. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 68:104–114, 1999.

[[San92]] Davide Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh, U. K., 1992.

[[San96]] Davide Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996.

[[San00]] Davide Sangiorgi. Lazy functions and mobile processes. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. M. I. T. Press, 2000.

[[SW01]] Davide Sangiorgi and David Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[[Vic94]] Björn Victor. *A Verification Tool for the Polyadic π-Calculus*. Licentiate thesis, Department of Computer Systems, Uppsala University, Sweden, 1994. Available as report DoCS 94/50.

[[VM94]] Björn Victor and Faron Moller. The Mobility Workbench — a tool for the π-calculus. In *Proceedings of the 6th International Conference on Computer Aided Verification (CAV '94)*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.

A. PROOFS OF RESULTS IN SECTION 3

Proposition A.1 *The relation $\sim_b$ is preserved by prefixing, restriction and choice operators.*

PROOF: Let P, Q, $R \in \mathcal{P}_S$ be such that $P \sim_b Q$, and let $u = x_1 \cdot ... \cdot x_n$ where $n \in \mathbb{N}$. Then:

- $\alpha.P \sim_b \alpha.Q$ since by applying the rule PREFIX we can conclude that $i)$ if $\alpha = u(y)$ or $\overline{u}\langle y\rangle$ we have that both $P \downarrow_u$ and $Q \downarrow_u$ or both $P \downarrow_{\overline{u}}$ and $Q \downarrow_{\overline{u}}$ respectively; $ii)$ if $\alpha = \tau$ we have that $\tau.P \xrightarrow{\tau} P$ and $\tau.Q \xrightarrow{\tau} Q$ and by hypothesis $P \sim_b Q$.
- $(\nu x)P \sim_b (\nu x)Q$ since by applying rules RES or OPEN we have that $(\nu x)P \downarrow_\beta$ if and only if $(\nu x)Q \downarrow_\beta$. In addition, if by applying RES $(\nu x)P \xrightarrow{\tau} (\nu x)P'$ then we had that $P \xrightarrow{\tau} P'$ and since $P \sim_b Q$ we would also have that $Q \xrightarrow{\tau} Q'$ and hence $(\nu x)Q \xrightarrow{\tau} (\nu x)Q'$ where $P' \sim_b Q'$ and so $(\nu x)P' \sim_b (\nu x)Q'$ as expected.
- $P + R \sim_b Q + R$ since $P + R$ exhibits a barb β if and only if so does P or R. Analogously, $Q + R$ exhibits a barb β if and only if so does Q or R. Since by hypothesis we have that $P \sim_b Q$, we can conclude that $P + R$ and $Q + R$ exhibit the same barbs.

□

Lemma A.2 *$P \simeq_b Q$ if and only if $P\sigma \sim_{beq} Q\sigma$ for any substitution σ*

PROOF:

($\Rightarrow$) Let P, $Q \in \mathcal{P}_S$ be such that $P \simeq_b Q$. Also, let $u = x_1 \cdot ... \cdot x_k$ where $k \in \mathbb{N}$ and $x_1, ..., x_k$ are fresh, and $\sigma = \{\tilde{y}/\tilde{z}\}$ be a substitution where $\tilde{y} = y_1, ..., y_n$ and $\tilde{z} = z_1, ..., z_n$.
Given $C = \overline{u}\langle y_1\rangle.\cdots.\overline{u}\langle y_n\rangle \mid u(z_1).\cdots.u(z_n).[\cdot] \mid R$ we know that $C[P] \sim_b C[Q]$ since both processes exhibit the same barbs. In addition, by performing n internal actions $C[P] \xrightarrow{\tau} ... \cdots \xrightarrow{\tau} P\sigma \mid R$, and the only process that does not exhibit barb $\overline{u}$ to which $C[Q]$ reduces in n steps is $Q\sigma \mid R$. Thus $P\sigma \mid R \sim_b Q\sigma \mid R$, that is, $P\sigma \sim_{beq} Q\sigma$.

($\Leftarrow$) Let P, $Q \in \mathcal{P}_S$ and σ be a substitution such that $P\sigma \sim_{beq} Q\sigma$, that is, $P\sigma \mid R \sim_b Q\sigma \mid R$ for any R. Since $\simeq_b$ is the largest congruence in $\sim_b$ it suffices to show that for any context C we have that $C[P]\sigma \mid R \sim_b C[Q]\sigma \mid R$. The proof is done by induction on C and we consider only the relevant transitions. Let $C = u(y).C'$ where $u = x_1 \cdot ... \cdot x_k$ and $k \in \mathbb{N}$.

- If by application of rules PREFIX and PAR1 one has $C[P]\sigma \mid R \xrightarrow{\sigma(u(y))} C'[P]\sigma \mid R$, then $C[Q]\sigma \mid R \xrightarrow{\sigma(u(y))} C'[Q]\sigma \mid R$ and by induction hypothesis $C'[P]\sigma \mid R \sim_b C'[Q]\sigma \mid R$.
- If by application of rule COMM one has $C[P]\sigma \mid R \xrightarrow{\tau} C'[P]\sigma\{z/y\} \mid R'$ (where we assume $R \xrightarrow{\overline{u}\langle z\rangle} R'$), then $C[Q]\sigma \mid R \xrightarrow{\tau} C'[Q]\sigma\{z/y\} \mid R'$ and by induction hypothesis $C'[P]\sigma\{z/y\} \mid R' \sim_b C'[Q]\sigma\{z/y\} \mid R'$.
- If by application of rule CLOSE one has $C[P]\sigma \mid R \xrightarrow{\tau} (\nu y)C'[P]\sigma \mid R'$ (where we assume $R \xrightarrow{\overline{u}(y)} R'$), then $C[Q]\sigma \mid R \xrightarrow{\tau} (\nu y)C'[Q]\sigma \mid R'$; by induction hypothesis $C'[P]\sigma \mid R' \sim_b C'[Q]\sigma \mid R'$ and by Proposition A.1 $\sim_b$ is closed under restriction, so $(\nu y)C'[P]\sigma \mid R' \sim_b (\nu y)C'[Q]\sigma \mid R'$.

The other cases can be handled in a similar way (for $C = !C'$ check reference [SW01]).

□

Lemma A.3 *The relation $\sim_g$ is preserved by the restriction operator.*

PROOF: Let us prove that $\mathcal{R} = \{((\nu x)P, (\nu x)Q) \mid P \sim_g Q\}$ is a ground bisimulation. We establish the proof by performing a case analysis on the rule used to infer an action for $(\nu x)P$, where we assume that $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P, Q) = \emptyset$.

- Case of rule RES where $x \notin \mathsf{nm}(\alpha)$. By definition of $\sim_g$, since $P \xrightarrow{\alpha} P'$ we have that $Q \xrightarrow{\alpha} Q'$ and $P' \sim_g Q'$. Therefore, by application of rule RES, $(\nu x)Q \xrightarrow{\alpha} (\nu x)Q'$ where $x \notin \mathsf{nm}(\alpha)$.

- Case of rule OPEN where $\overline{u} = \overline{x_1 \cdot ... \cdot x_k}$ for some $k \in \mathbb{N}$ and $x \notin \mathsf{nm}(\overline{u})$. By definition of $\sim_g$, since $P \xrightarrow{\overline{u}\langle x\rangle} P'$ we have that $Q \xrightarrow{\overline{u}\langle x\rangle} Q'$ and $P' \sim_g Q'$. Therefore, by application of rule OPEN, $(\nu x)Q \xrightarrow{\overline{u}(x)} Q'$ where $x \notin \mathsf{nm}(\overline{u})$.

Therefore, $\mathcal{R}$ is a ground bisimulation.

□

Proposition A.4 *The relation $\sim_g$ is preserved by prefixing and choice.*

PROOF: The proof follows the pattern of the one above.

□

Recall that every subset of a countable set is again countable, and that the countable union of countable sets is countable. Note that we consider a set to be *countable* if it is either finite or has the same cardinality as the set of the natural numbers $\mathbb{N}$.

Theorem A.5 $\sim_e = \sim_{beq}$

PROOF:

(⇒) We prove that $\sim_e$ is a barbed bisimulation. Let $u = x_1 \cdot ... \cdot x_n$ where $n \in \mathbb{N}$ and let $P, Q \in \mathcal{P}_S$ be such that $P\mathcal{S}Q$ where $\mathcal{S}$ is an early bisimulation. Then, if:

1. $P \xrightarrow{u(y)} P'$ then $P \downarrow_u$ and since P and Q are early bisimilar for each w there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and $P'\{w/y\}\mathcal{S}Q'\{w/y\}$. Thus, since $Q \xrightarrow{u(y)} Q'$, we have that $Q \downarrow_u$.
2. $P \xrightarrow{\alpha} P'$ where $\alpha = \overline{u}\langle y\rangle$ or $\overline{u}(y)$ then $P \downarrow_{\overline{u}}$ and since P and Q are early bisimilar there is a Q' such that $Q \xrightarrow{\alpha} Q'$ which implies that $Q \downarrow_{\overline{u}}$.
3. $P \xrightarrow{\tau} P'$ since P and Q are early bisimilar there is a Q' such that $Q \xrightarrow{\tau} Q'$ (and $P'\mathcal{S}Q'$). Note that to P' and Q' we can apply the same reasoning as to the P and Q we started from till a visible action is performed by both processes (or their descendants) like in case 1 or 2, or until both processes are inactive.

Therefore, we can conclude that P and Q are barbed bisimilar. Since early bisimulation is preserved by all operators except input prefixing (Lemma 3.18), we have that for any R, $P \mid R \sim_e Q \mid R$, and thus $P \mid R \sim_b Q \mid R$. We then comply with the necessary requirements of barbed equivalence and establish that $P \sim_{beq} Q$.

(⇐) Given $\mathcal{S} = \{(P, Q) : F, Y, \tilde{x}$ *exist such that* $(\nu\tilde{x})C^{F,Y}[P] \sim_b (\nu\tilde{x})C^{F,Y}[Q]\}$ where $\tilde{x}$, F, $H\langle F\rangle$, Y are related as explained before, we prove that $\mathcal{S}$ is an early bisimulation.
The context $C[\cdot]$ is defined as $C[\cdot] = [\cdot] \mid V\langle F, Y\rangle$ where:
$V\langle F, Y\rangle =$

$$\sum_{(c,c'')\in H\langle F\rangle} \sum_{(b,b')\in F\cup(y,y')} \overline{c}\langle b\rangle .(c'' + b' + in + V\langle F \cup (y, y'), Y\backslash(y, y')\rangle) \tag{3}$$

$$+ \sum_{(c,c'')\in H\langle F\rangle} c(y).(c'' + y' + out + V\langle F \cup (y, y'), Y\backslash(y, y')\rangle + (\nu t) \sum_{(b,b')\in F} (\overline{t \cdot b} \mid t \cdot y).b' \tag{4}$$

The first (3) and second (4) summands are used to test respectively the input and output actions of P or Q[12]. Note in 3 that all possible inputs are considered in the inner summation just like early bisimulation requires. In 4 the last term in the summation concerns the case of bound output in which the outputted name will not be found in H, as opposed to the case of free output. Further, notice that the names in, out are not in $\mathsf{nm}(P, Q)$, these names are used to show which type of action (input or output, respectively) was performed.
The relation between F and Y can now be further analysed, since the names taken from Y are used

[12]Note that the summations in (3) and (4) are finite.

to augment F (and hence H) via name-communication. Note that on the definition of $V\langle F,Y\rangle$ the pair (y,y') is drawn from Y.
We now prove the core of the theorem by splitting the proof into the four possible actions of P.

1. If $P \xrightarrow{\tau} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P \mid V\langle F,Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(P' \mid V\langle F,Y\rangle) = R$. Since by hypothesis $(\nu\tilde{x})C^{F,Y}[P] \sim_b (\nu\tilde{x})C^{F,Y}[Q]\}$, then $(\nu\tilde{x})C^{F,Y}[Q] \xrightarrow{\tau} T$, and R may have as barbs only c such that $c \in H_1$, and so should T. Note that $(\nu\tilde{x})C^{F,Y}[Q]$ could have performed a τ action by $i)$ interaction between $V\langle F,Y\rangle$ and process Q where Q performed an input action; $ii)$ interaction between $V\langle F,Y\rangle$ and process Q where Q performed a free or bound output action; $iii)$ there is no interaction with $V\langle F,Y\rangle$ and Q performs a τ action by itself. Both $i)$ and $ii)$ are impossible since at least $T \downarrow_{in}$ or $T \downarrow_{out}$ and $\{in,out\} \cap H_1 = \emptyset$. Thus, $(\nu\tilde{x})C^{F,Y}[Q] \xrightarrow{\tau} T = (\nu\tilde{x})(Q' \mid V\langle F,Y\rangle)$, that is $Q \xrightarrow{\tau} Q'$ and $(P',Q') \in \mathcal{S}$.
2. If $P \xrightarrow{c(y)} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P \mid V\langle F,Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(P'\{b/y\} \mid V_1) = R$ where $V_1 = c'' + b' + in + V\langle F \cup (y,y'), Y\backslash(y,y')\rangle$. Then, R has as barbs at least c'', b', and in. Note that $(\nu\tilde{x})C^{F,Y}[Q]$ could have performed a τ action by the $i)$, $ii)$ and $iii)$ reasons mentioned in case 1. The situation $ii)$ where Q performed a free or bound output is impossible since in that case $T \downarrow_{out}$ but $R \not\downarrow_{out}$. The situation $iii)$ is impossible since in that case, e.g., $T \not\downarrow_{b'}$. The only possible situation is then if Q performed an input action of the type $\alpha = c_1(b_1)$. Note that the following equalities have to hold $c_1 = c$ and $b_1 = b$ so that $T \downarrow_{c''}$ and $T \downarrow_{b'}$ as does R. Thus, we have that if $P \xrightarrow{c(y)} P'$ then for every possible b we have that $Q \xrightarrow{c(y)} Q'$ and $R \sim_b T$, that is, $(P'\{b/y\}, Q'\{b/y\}) \in \mathcal{S}$.
3. If $P \xrightarrow{\overline{c}\langle z\rangle} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P \mid V\langle F,Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(P' \mid V_2) = R$ where $V_2 = c'' + z' + out + V\langle F,Y\rangle + (\nu t)\sum_{(b,b')\in F}(\overline{t \cdot b} \mid t \cdot y).b'$ and t is fresh, while $z \in \mathsf{fn}(P) \subseteq F_1$. Then, R has as barbs at least c'', z' and out. Since by hypothesis $(\nu\tilde{x})C^{F,Y}[P] \sim_b (\nu\tilde{x})C^{F,Y}[Q]\}$, then $(\nu\tilde{x})C^{F,Y}[Q] \xrightarrow{\tau} T$ and T should have the same barbs as R. Note that $(\nu\tilde{x})C^{F,Y}[Q]$ could have performed a τ action by the $i)$, $ii)$ and $iii)$ reasons mentioned in case 1. The situation $i)$ is impossible since $T \downarrow_{in}$ but $R \not\downarrow_{in}$, and the situation $iii)$ is impossible since, e.g., $T \not\downarrow_{c''}$. Then we are in a situation where Q has to do an output on the same channel as P (so it has c'' as a barb too), but we must still prove the output has to be free. This happens because since P performed a free output, R can do a τ action from the last summation in V_2 and the resulting process has as an unique barb z'. However, if Q performs a bound output, the summation in V_2 is an inactive process, and even if $(\nu\tilde{x})C^{F,Y}[Q]$ performs a τ action by the cases $i)$, $ii)$ and $iii)$, in both $ii)$ and $iii)$ the resulting process would have either in or out as a barb. In the case of $i)$ then there would be no interaction with $V\langle F,Y\rangle$, and so the resulting process would not have z' as a barb. Thus, Q has to perform $\alpha = \overline{c}\langle z\rangle$ as did P.
4. If $P \xrightarrow{\overline{c}(y)} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P \mid V\langle F,Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(\nu y)(P' \mid V_3) = R$ where $V_3 = c'' + y' + out + V\langle F \cup (y,y'), Y\backslash(y,y')\rangle + (\nu t)\sum_{(b,b')\in F}(\overline{t \cdot b} \mid t \cdot y).b'$. The rest of the proof is very similar to case 3 where the difference between bound and free output is analysed.

□

B. PROOFS OF RESULTS IN SECTION 5

Lemma B.1 *Substitution Lemma*
$[\![P\sigma]\!] = [\![P]\!]\sigma$, *for any substitution* σ.

PROOF: By induction on the structure of crypto-epi processes. Since the relevant cases are those of the new constructs (as the encoding is homomorphic in the other), we only analyse these cases.

Case $P = \mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, P'$; consider $\sigma = \{m'/m\}$[13], thus $P\sigma = \mathsf{encrypt}\, m' \twoheadrightarrow^k x \,\mathsf{in}\, P'\{m'/m\}$. Since by induction hypothesis, $[\![P'\{m'/m\}]\!] = [\![P']\!]\{m'/m\}$, we proceed as below.

$$\begin{array}{rcl} [\![P\sigma]\!] & = & (\nu x)(\overline{x \cdot k}\langle m'\rangle \mid [\![P'\{m'/m\}]\!]) \\ & = & (\nu x)(\overline{x \cdot k}\langle m'\rangle \mid [\![P']\!]\{m'/m\}) \\ & = & ((\nu x)(\overline{x \cdot k}\langle m\rangle \mid [\![P']\!]))\{m'/m\} \\ & = & [\![P]\!]\sigma \end{array}$$

Case $P = \mathsf{decrypt}\, x \twoheadrightarrow^k y \,\mathsf{in}\, P'$; consider $\sigma = \{m'/m\}$[14], thus $P\sigma = \mathsf{decrypt}\, x' \twoheadrightarrow^k y \,\mathsf{in}\, P'\{x'/x\}$. Since by induction hypothesis, $[\![P'\{m'/m\}]\!] = [\![P']\!]\{m'/m\}$, we proceed as below.

$$\begin{array}{rcl} [\![P\sigma]\!] & = & x' \cdot k(y).[\![P'\{x'/x\}]\!] \\ & = & x' \cdot k(y).[\![P']\!]\{x'/x\} \\ & = & (x \cdot k(y).[\![P']\!])\{x'/x\} \\ & = & [\![P]\!]\sigma \end{array}$$

□

Lemma B.2

1. *If* $[|P|] \xrightarrow{\overline{u}y} Q$ *then, for some* P_1, P_2 *and some* $\tilde{n}$ *such that* $n(u) \cap \{\tilde{n}\} = \emptyset$, *either*
   (a) $P \equiv (\nu\tilde{n})(\overline{u}y.P_1|P_2)$ *and* $Q \equiv (\nu\tilde{n})([|P_1|] \mid [|P_2|])$; *or*
   (b) $P \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, \overline{u}y.P_1|P_2)$ *where* $x \neq y$, *and* $Q \equiv (\nu\tilde{n})(\overline{x \cdot k}m \mid [|P_1|] \mid [|P_2|])$.
2. *If* $[|P|] \xrightarrow{\overline{u}(y)} Q$ *then for some* P_1, P_2 *and some* $\tilde{n}$ *such that* $n(u) \cap \{\tilde{n}\} = \emptyset$, *if* $y \neq x$, *then either*
   (a) $P \equiv (\nu\tilde{n}, y)(\overline{u}y.P_1|P_2)$ *and* $Q \equiv (\nu\tilde{n})([|P_1|] \mid [|P_2|])$; *or*
   (b) $P \equiv (\nu\tilde{n}, y)(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, \overline{u}y.P_1|P_2)$ *and* $Q \equiv (\nu\tilde{n}, x)(\overline{x \cdot k}m \mid [|P_1|] \mid [|P_2|])$;

   otherwise, if $x = y$ *then*
   $P \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k y \,\mathsf{in}\, \overline{u}y.P_1|P_2)$ *and*
   $Q \equiv (\nu\tilde{n})(\overline{y \cdot k}m \mid [|P_1|] \mid [|P_2|])$.
3. *If* $[|P|] \xrightarrow{u(y)} Q$ *then for some* P_1, P_2, *if for some* $\tilde{n}$ *such that* $n(u) \cap \{\tilde{n}\} = \emptyset$, *then either*
   (a) $P \equiv (\nu\tilde{n})(u(y).P_1|P_2)$ *and* $Q \equiv (\nu\tilde{n})([|P_1|] \mid [|P_2|])$; *or*
   (b) $P \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, u(y).P_1|P_2)$ *and* $Q \equiv (\nu\tilde{n}, x)(\overline{x \cdot k}m \mid [|P_1|] \mid [|P_2|])$;

   otherwise, if $u = x \cdot k$ *then*
   $P \equiv (\nu\tilde{n})(\mathsf{decrypt}\, x \twoheadrightarrow^k y \,\mathsf{in}\, P_1|P_2)$, *and*
   $Q \equiv (\nu\tilde{n})([|P_1|] \mid [|P_2|])$.

PROOF: Follows directly from the definition of processes in the cryptographic π-calculus with polyadic synchronization and from the transition rules.

□

Lemma B.3 *Operational Correspondence*

1. *If* $[\![P]\!] \xrightarrow{\alpha} Q$ *then there is a* P' *such that* $P \xrightarrow{\alpha} P'$ *and* $[|P'|] = Q$.
2. *If* $P \xrightarrow{\alpha} P'$ *then* $[|P|] \xrightarrow{\alpha} [|P'|]$.

PROOF: The proof is done by induction on the inference of the transition of $[\![P]\!] \xrightarrow{\alpha} Q$.

1. Let $\alpha = \overline{u}y$. By Lemma B.2 we have that:
   (a) $P \equiv (\nu\tilde{n})(\overline{u}y.P_1|P_2)$ in which case $P \xrightarrow{\overline{u}y} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.

[13]The other cases are handled in an analogous way. Note that if e.g. $\sigma = \{x/m\}$ then we would have to perform α-conversion.
[14]Again ,the other cases are handled in an analogous way.

(b) $P \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, \overline{u}y.P_1|P_2)$ in which case
$P \xrightarrow{\overline{u}y} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.

2. Let $\alpha = \overline{u}(y)$. By Lemma B.2 we have that:

(a) $P \equiv (\nu\tilde{n}, y)(\overline{u}y.P_1|P_2)$ in which case $P \xrightarrow{\overline{u}(y)} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.
(b) $P \equiv (\nu\tilde{n}, y)(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, \overline{u}y.P_1|P_2)$ in which case
$P \xrightarrow{\overline{u}(y)} P' \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, P_1|P_2)$ and $[|P'|] = Q$.
(c) $P \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k y \,\mathsf{in}\, \overline{u}y.P_1|P_2)$ in which case
$P \xrightarrow{\overline{u}(y)} P' \equiv (\nu\tilde{n})(\overline{y \cdot k}m|P_1|P_2)$ and $[|P'|] = Q$.

3. Let $\alpha = u(y)$. By Lemma B.2 we have that:

(a) $P \equiv (\nu\tilde{n})(u(y).P_1|P_2)$ in which case $P \xrightarrow{u(y)} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.
(b) $P \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, u(y).P_1|P_2)$ in which case
$P \xrightarrow{u(y)} P' \equiv (\nu\tilde{n})(\mathsf{encrypt}\, m \twoheadrightarrow^k x \,\mathsf{in}\, P_1|P_2)$ and $[|P'|] = Q$.
(c) $u = x \cdot k$ and $P \equiv (\nu\tilde{n})(\mathsf{decrypt}\, x \twoheadrightarrow^k y \,\mathsf{in}\, P_1|P_2)$ in which case
$P \xrightarrow{x \cdot k(y)} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.

4. Let $\alpha = \tau$. The most relevant cases result from the application of rules COMM or CLOSE. Let us analyse the case of rule COMM1 (the remaining cases follow in a similar manner).
Consider $P \xrightarrow{\overline{u}y} P'$ and $Q \xrightarrow{u(z)} Q'$. Then $P|Q \xrightarrow{\tau} P'|Q'\{y/z\}$, and by clauses 1 and 3 of this lemma, there exist P_1, P_1', Q_1, Q_1' such that $[|P_1|] = P$, $[|Q_1|] = Q$, $P_1 \xrightarrow{\overline{u}y} P_1'$ where $[|P_1'|] = P'$, and $Q_1 \xrightarrow{u(z)} Q_1'$ where $[|Q_1'|] = Q'$. Thus, $P_1|Q_1 \xrightarrow{\tau} P_1'|Q_1'\{y/z\}$, and using Lemma B.1, we conclude as needed.

$$[|P_1'|Q_1'\{y/z\}|] = [|P_1'|] \,|\, [|Q_1'\{y/z\}|] = [|P_1'|] \,|\, [|Q_1'|]\{y/z\} = P'|Q'\{y/z\}$$

□

Lemma B.4

1. *If* $[\![P]\!] \sim_e [\![Q]\!]$ *then* $P \sim_e Q$.
2. *If* $[\![P]\!] \sim_{beq} [\![Q]\!]$ *then* $P \sim_{beq} Q$.

PROOF:

1. We prove that $\mathcal{R} = \{(P, Q) : [\![P]\!] \sim_e [\![Q]\!]\}$ is an early bisimulation (*cf.* Definition 3.16 in page 9).

Case $P \xrightarrow{\alpha} P'$ (the case $Q \xrightarrow{\alpha} Q'$ is similar, and we omit its analysis).
Then, by Lemma 5.5.2 we have that $[\![P]\!] \xrightarrow{\alpha} [\![P']\!]$. Since by hypothesis $[\![P]\!] \sim_e [\![Q]\!]$ then there is a Q' such that $[\![Q]\!] \xrightarrow{\alpha} Q'$, and by Lemma 5.5.1 we have that there is a Q'' such that $Q \xrightarrow{\alpha} Q''$, where $[\![Q'']\!] = Q'$.

We now split the proof according to the possible transitions of $[\![P]\!]$.

Case $\alpha \in \{\tau, \overline{u}y, \overline{u}(y)\}$, where $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P, Q) = \emptyset$.
By definition of $\sim_e$ we have that $[\![P']\!] \sim_e [\![Q'']\!]$ and therefore $P' \mathcal{R} Q''$.

Case $\alpha = u(y)$ where $y \notin \mathsf{fn}(P, Q)$. The reasoning is similar to the one above.
By definition of $\sim_e$ we have that $[\![P']\!]\{w/y\} \sim_e [\![Q'']\!]\{w/y\}$ and by application of Lemma B.1 we know that $[\![P'\{w/y\}]\!] \sim_e [\![Q''\{w/y\}]\!]$. Therefore, we conclude that $P'\{w/y\} \mathcal{R} Q''\{w/y\}$.

2. Follows directly from Lemma B.4.1 and Theorem A.5, where it was established that early bisimulation coincides with barbed equivalence.

□

C. PROOFS OF RESULTS IN SECTION 4

Proposition C.1

$$[|\,\mathsf{encrypt}\, m \looparrowright^k x \,\mathsf{in}\, (\mathsf{decrypt}\, x \looparrowright^k y \,\mathsf{in}\, (\mathsf{if}\ m = y\ \mathsf{then}\ m().\mathbf{0}\,))\,|] \ \simeq_e\ m().\mathbf{0}$$

PROOF: By definition,

$$[|\,\mathsf{encrypt}\, m \looparrowright^k x \,\mathsf{in}\, (\mathsf{decrypt}\, x \looparrowright^k y \,\mathsf{in}\, (\mathsf{if}\ m = y\ \mathsf{then}\ m().\mathbf{0}\,))\,|] \stackrel{\text{def}}{=} (\nu x)(!\overline{x \cdot k}\langle m\rangle \,|x{\cdot}k(y).(\nu z)(\overline{z \cdot m}\langle\rangle \,|z{\cdot}y().m().\mathbf{0}\,)$$

Construct the appropriate bisimulation, noticing that in two deterministic tau-steps one gets

$$(\nu x)(!\overline{x \cdot k}\langle m\rangle \ |\ (\nu z)(m().\mathbf{0}\,) \simeq_e\ m().\mathbf{0}$$

□

The proof of Proposition 4.2 is similar, requiring five deterministic tau-steps.

(PREFIX) $$\frac{-}{\alpha.P \xrightarrow{\alpha} P}$$

(CH1) $$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

(PAR1) $$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \quad \text{where } \mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \emptyset$$

(RES) $$\frac{P \xrightarrow{\alpha} P'}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'} \quad \text{where } x \notin \mathsf{nm}(\alpha)$$

(REP-ACT) $$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'|!P}$$

(REP-COMM) $$\frac{P \xrightarrow{\overline{u}\langle x\rangle} P' \quad P \xrightarrow{u(z)} P''}{!P \xrightarrow{\tau} (P'|P''\{x/z\})|!P}$$

(REP-CLOSE) $$\frac{P \xrightarrow{\overline{u}(x)} P' \quad P \xrightarrow{u(x)} P''}{!P \xrightarrow{\tau} (\nu x)(P'|P'')|!P} \quad \text{where } x \notin \mathsf{fn}(P)$$

(OPEN) $$\frac{P \xrightarrow{\overline{u}\langle x\rangle} P'}{(\nu y)P \xrightarrow{\overline{u}(x)} P'} \quad \text{where } x \notin \mathsf{nm}(\overline{u})$$

(CLOSE1) $$\frac{P \xrightarrow{\overline{u}(x)} P' \quad Q \xrightarrow{u(x)} Q'}{P|Q \xrightarrow{\tau} (\nu x)(P'|Q')}$$

(COMM1) $$\frac{P \xrightarrow{\overline{u}\langle x\rangle} P' \quad Q \xrightarrow{u(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{x/z\}}$$

(CONV) $$\frac{P \xrightarrow{\alpha} P'}{Q \xrightarrow{\alpha} P'} \quad \text{if } Q =_\alpha P$$

FIGURE 2: Late transition rules.

$$\text{(ENC)} \quad \frac{P \xrightarrow{\alpha} P'}{\mathsf{encrypt}\, m \looparrowright^{k} x \,\mathsf{in}\, P \xrightarrow{\alpha} \mathsf{encrypt}\, m \looparrowright^{k} x \,\mathsf{in}\, P'}$$

where $\alpha \neq \overline{u}\langle x \rangle$ and if $\alpha \in \{\overline{u}\langle y \rangle, \overline{u}(y), u(y)\}$ then $x \notin \mathsf{nm}(u)$

$$\text{(ENC-OPEN)} \quad \frac{P \xrightarrow{\overline{u}\langle x \rangle} P'}{\mathsf{encrypt}\, m \looparrowright^{k} x \,\mathsf{in}\, P \xrightarrow{\overline{u}(x)} !\overline{x \cdot k}\langle m \rangle \mid P'}$$

where $x \notin \mathsf{nm}(u)$

$$\text{(DEC)} \quad \frac{--}{\mathsf{decrypt}\, x \looparrowright^{k} y \,\mathsf{in}\, P \xrightarrow{x \cdot k(y)} P}$$

FIGURE 3: Late transition rules for the cryptographic constructs