Information and Computation **148**, 1–70 (1999)
Article ID inco.1998.2740, available online at http://www.idealibrary.com on IDEAL®

A Calculus for Cryptographic Protocols: The Spi Calculus

Martín Abadi

Systems Research Center, Digital Equipment Corporation, Palo Alto, California

and

Andrew D. Gordon*

Computer Laboratory, University of Cambridge, Cambridge, United Kingdom

We introduce the spi calculus, an extension of the pi calculus designed for describing and analyzing cryptographic protocols. We show how to use the spi calculus, particularly for studying authentication protocols. The pi calculus (without extension) suffices for some abstract protocols; the spi calculus enables us to consider cryptographic issues in more detail. We represent protocols as processes in the spi calculus and state their security properties in terms of coarse-grained notions of protocol equivalence.

1. SECURITY AND THE PI CALCULUS

The spi calculus is an extension of the pi calculus [MPW92] with cryptographic primitives. It is designed for describing and analyzing security protocols, such as those for authentication and for electronic commerce. These protocols rely on cryptography and on communication channels with properties like authenticity and privacy. Accordingly, cryptographic operations and communication through channels are the main ingredients of the spi calculus.

We use the pi calculus (without extension) for describing protocols at an abstract level. The pi calculus primitives for channels are simple but powerful. Channels can be created and passed, for example from authentication servers to clients. The scoping rules of the pi calculus guarantee that the environment of a protocol (the attacker) cannot access a channel that it is not explicitly given; scoping is thus the basis of security. In sum, the pi calculus appears as a fairly convenient calculus of protocols for secure communication.

However, the pi calculus does not express the cryptographic operations that are commonly used for implementing channels in distributed systems: it does not

* Currently at Microsoft Research, Cambridge.

0890-5401/99 $30.00

include any constructs for encryption and decryption, and these do not seem easy to represent. Since the use of cryptography is notoriously error prone, we prefer not to abstract it away. We define the spi calculus in order to permit an explicit representation of the use of cryptography in protocols.

There are by now many other notations for describing security protocols. Some, which have long been used in the authentication literature, have a fairly clear connection to the intended implementations of those protocols (see, e.g., [NS78, Lie93]). Their main shortcoming is that they do not provide a precise and solid basis for reasoning about protocols. Other notations (e.g., [BAN89]) are more formal, but their relation to implementations may be more tenuous or subtle. The spi calculus is a middle ground: it is directly executable and it has a precise semantics.

Because the semantics of the spi calculus is not only precise but intelligible, the spi calculus provides a setting for analyzing protocols. Specifically, we can express security guarantees as equivalences between spi calculus processes. For example, we can say that a protocol keeps secret a piece of data X by stating that the protocol with X is equivalent to the protocol with X', for every X'. Here, equivalence means equivalence in the eyes of an arbitrary environment. The environment can interact with the protocol, perhaps attempting to create confusion between different messages or sessions. This definition of equivalence yields the desired properties for our security applications. (Interestingly, we cannot take the standard bisimilarity relation as our notion of equivalence.) Moreover, equivalence is not too hard to prove; we demonstrate this by carrying out the analysis of a few small protocols.

Although the definition of equivalence makes reference to the environment, we do not need to give a model of the environment explicitly. This is one of the main advantages of our approach. Writing such a model can be tedious and can lead to new arbitrariness and error. In particular, it is always difficult to express that the environment can invent random numbers but is not lucky enough to guess the random secrets on which a protocol depends. We resolve this conflict by letting the environment be an arbitrary spi calculus process.

Our approach has some similarities with other recent approaches for reasoning about protocols. Like work based on temporal logics or process algebras (e.g., [GM95, Low96, Sch96a]), our method builds on a standard concurrency formalism; this has obvious advantages but it also implies that our method is less intuitive than some based on ad hoc formalisms (e.g., [BAN89]). As in some modal logics (e.g., [ABLP93, LABW92]), we emphasize reasoning about channels and their utterances. As in state-transition models (e.g., [DY81, MCF87, Mil95a, Kem89, Mea92, Pau97]), we are interested in characterizing the knowledge of an environment. The unique features of our approach are its reliance on the powerful scoping constructs of the pi calculus, the radical definition of the environment as an arbitrary spi calculus process, and the representation of security properties, both integrity and secrecy, as equivalences.

Our model of protocols is simpler, but poorer, than some models developed for informal mathematical arguments (e.g., [BR95]) because the spi calculus does not include any notion of probability or complexity. It would be interesting to bridge the gap between the spi calculus and those models, perhaps by giving a probabilistic interpretation for our results.

Contents of this Paper

Section 2 introduces the pi calculus and our method of specifying authenticity and secrecy properties as equations. Section 3 extends the pi calculus with primitives for shared-key cryptography. Sections 4 and 5 define the formal semantics of the spi calculus and associated proof techniques, respectively. Section 6 uses these techniques in proofs of some of the properties stated earlier. Section 7 discusses how to add primitives for public-key cryptography to the pi calculus, and Section 8 offers some conclusions. The Appendices contain some proofs; a technical report [AG97a] contains additional proofs, as well as sketches of partial encodings of the spi calculus in the pi calculus.

Two conference papers contain part of the material of this paper, in preliminary form [AG97b, AG97c]. Other recent papers describe additional proof techniques [AG98] and a type system [Aba97] for the spi calculus.

2. PROTOCOLS USING RESTRICTED CHANNELS

In this section, we review the definition of the pi calculus informally. (We give a more formal presentation in Section 4.) We then introduce a new application of the pi calculus, namely its use for the study of security.

2.1. Basics

The pi calculus is a small but extremely expressive programming language. It is an important result of the search for a calculus that could serve as a foundation for concurrent computation, in the same way in which the lambda calculus is a foundation for sequential computation.

Pi calculus programs are systems of independent, parallel processes that synchronize via message-passing handshakes on named channels. The channels that a process knows about determine the communication possibilities of the process. Channels may be *restricted*, so that only certain processes may communicate on them. In this respect the pi calculus is similar to earlier process calculi such as CSP [Hoa85] and CCS [Mil89].

What sets the pi calculus apart from earlier calculi is that the scope of a restriction—the program text in which a channel may be used—may change during computation. When a process sends a restricted channel as a message to a process outside the scope of the restriction, the scope is said to *extrude*, that is, it enlarges to embrace the process receiving the channel. Processes in the pi calculus are mobile in the sense that their communication possibilities may change over time; they may learn the names of new channels via scope extrusion. Thus, a channel is a transferable capability for communication.

A central technical idea of this paper is to use the restriction operator and scope extrusion from the pi calculus as a formal model of the possession and communication

of secrets, such as cryptographic keys. These features of the pi calculus are essential in our descriptions of security protocols.

2.2. Outline of the Pi Calculus

There are in fact several versions of the pi calculus. Here we present the syntax and semantics of a particular version of the pi calculus; although this version is not the standard one, our choices should be relatively uncontroversial. The differences with other versions are mostly orthogonal to our concerns.

We assume an infinite set of *names*, to be used for communication channels, and an infinite set of *variables*. We let m, n, p, q, and r range over names, and let x, y, and z range over variables.

The set of *terms* is defined by the grammar:

$L, M, N ::=$	terms
n	name
(M, N)	pair
0	zero
$suc(M)$	successor
x	variable

In the standard pi calculus, names are the only terms. For convenience we have added constructs for pairing and numbers, namely (M, N), 0, and $suc(M)$, and we have also distinguished variables from names. (This distinction simplifies the treatment of some equivalences.)

The set of processes is defined by the grammar:

$P, Q, R ::=$	processes
$\bar{M}\langle N\rangle.P$	output
$M(x).P$	input
$P \mid Q$	composition
$(\nu n)\, P$	restriction
$!\, P$	replication
$[\, M \textit{ is } N\,]\, P$	match
$\mathbf{0}$	nil
$\textit{let } (x, y) = M \textit{ in } P$	pair splitting
$\textit{case } M \textit{ of } 0: P \; suc(x): Q$	integer case

In $(\nu n)\, P$, the name n is bound in P. In $M(x).P$, the variable x is bound in P. In $\textit{let } (x, y) = M \textit{ in } P$, the variables x and y are bound in P. In $\textit{case } M \textit{ of } 0: P \; suc(x): Q$, the variable x is bound in the second branch, Q. We write $P[M/x]$ for the outcome of replacing each free occurrence of x in process P with the term M, and identify processes up to renaming of bound variables and names. We adopt the abbreviation $\bar{M}\langle N\rangle$ for $\bar{M}\langle N\rangle.\mathbf{0}$.

Intuitively, the constructs of the pi calculus have the following meanings:

- The basic computational step and synchronization mechanism in the pi calculus is *interaction*, in which a term N is communicated from an output process to an input process via a named channel, m.

  — An *output process* $\bar{m}\langle N\rangle.P$ is ready to output on channel m. If an interaction occurs, term N is communicated on m and then process P runs.

  — An *input process* $m(x).P$ is ready to input from channel m. If an interaction occurs in which N is communicated on m, then process $P[N/x]$ runs.

(The general forms $\bar{M}\langle N\rangle.P$ and $M(x).P$ of output and input allow for the channel to be an arbitrary term M. The only useful cases are for M to be a name, or a variable that gets instantiated to a name.)

- A *composition* $P \mid Q$ behaves as processes P and Q running in parallel. Each may interact with the other on channels known to both, or with the outside world, independently of the other.

- A *restriction* $(\nu n)\ P$ is a process that makes a new, private name n, and then behaves as P.

- A *replication* $!P$ behaves as an infinite number of copies of P running in parallel.

- A *match* $[M \textit{ is } N]\ P$ behaves as P provided that terms M and N are the same; otherwise it is stuck, that is, it does nothing.

- The *nil* process $\mathbf{0}$ does nothing.

Since we added pairs and integers, we have two new process forms:

- A *pair splitting* process $\textit{let } (x, y) = M \textit{ in } P$ behaves as $P[N/x][L/y]$ if term M is the pair (N, L). Otherwise, the process is stuck.

- An *integer case* process $\textit{case } M \textit{ of } 0: P\ \textit{suc}(x): Q$ behaves as P if term M is 0, as $Q[N/x]$ if M is $\textit{suc}(N)$. Otherwise, the process is stuck.

We write $P \simeq Q$ to mean that the behaviours of the processes P and Q are indistinguishable. In other words, the processes P and Q may have different internal structure, but a third process R cannot distinguish running in parallel with P from running in parallel with Q. As far as R can tell, P and Q have the same properties (more precisely, the same safety properties). We define the relation $\simeq$ in Section 4.2 as a form of testing equivalence. For now, it suffices to understand $\simeq$ informally.

2.3. Examples Using Restricted Channels

Next, we show how to express some abstract security protocols in the pi calculus. In security protocols, it is common to find channels on which only a given set of principals is allowed to send data or to listen. The set of principals may expand in the course of a protocol run, for example as the result of channel establishment. Remarkably, it is easy to model this property of channels in the pi calculus, via the

restriction operation; the expansion of the set of principals that can access a channel corresponds to scope extrusion.

We do not provide a systematic translation from another language for describing protocols into the pi calculus, but rather show some examples of protocols written directly in the pi calculus, along with informal descriptions of the kind commonly found in the security literature. We do introduce a fairly systematic approach for stating properties of protocols as pi calculus equivalences.

2.3.1. *A first example.* Our first example is extremely basic. In this example, there are two principals A and B that share a channel, c_{AB}; only A and B can send data or listen on this channel. The protocol is simply that A uses c_{AB} for sending a single message M to B.

In informal notation, we may write this protocol as follows:

$$\text{Message 1} \quad A \to B\colon M \qquad \text{on } c_{AB}$$

A first pi calculus description of this protocol is:

$$A(M) \triangleq \overline{c_{AB}}\langle M\rangle$$
$$B \triangleq c_{AB}(x).\mathbf{0}$$
$$Inst(M) \triangleq (\nu c_{AB})(A(M) \mid B)$$

The processes $A(M)$ and B describe the two principals, and $Inst(M)$ describes (one instance of) the whole protocol. The channel c_{AB} is restricted; intuitively, this achieves the effect that only A and B have access to c_{AB}.

In these definitions, $A(M)$ and $Inst(M)$ are processes parameterized by M. More formally, we say that A and $Inst$ are abstractions, and treat the M's on the left of $\triangleq$ as bound parameters. Roughly, abstractions are functions that map terms to processes. (Section 5.1 contains a precise definition of abstractions.) Abstractions can of course be instantiated (applied); for example, the instantiation $A(0)$ yields $\overline{c_{AB}}\langle 0\rangle$. The standard rules of substitution govern application, forbidding parameter captures; for example, expanding $Inst(c_{AB})$ would require a renaming of the bound occurrence of c_{AB} in the definition of $Inst$.

The first pi calculus description of the protocol may seem a little futile because, according to it, B does nothing with its input. A more useful and general description says that B runs a process F with its input. We revise our definitions as

$$A(M) \triangleq \overline{c_{AB}}\langle M\rangle$$
$$B \triangleq c_{AB}(x).F(x)$$
$$Inst(M) \triangleq (\nu c_{AB})(A(M) \mid B)$$

Informally, $F(x)$ is simply the result of applying F to x. More formally, F is an abstraction, and $F(x)$ is an instantiation of the abstraction. We adopt the convention that the bound parameters of the protocol (in this case, M, c_{AB}, and x) cannot occur free in F.

This protocol has two important properties:

- Authenticity (or integrity): B always applies F to the message M that A sends; an attacker cannot cause B to apply F to some other message.
- Secrecy: The message M cannot be read in transit from A to B: if F does not reveal M, then the whole protocol does not reveal M.

The secrecy property can be stated in terms of equivalences: if $F(M) \simeq F(M')$, for all M and M', then $Inst(M) \simeq Inst(M')$. This means that if $F(M)$ is indistinguishable from $F(M')$, then the protocol with message M is indistinguishable from the protocol with message M'.

There are many sensible ways of formalizing the authenticity property. In particular, it may be possible to use notions of refinement or a suitable program logic. However, we choose to write authenticity as an equivalence, for economy. This equivalence compares the protocol with another protocol. Our intent is that the latter protocol serves as a specification. In this case, the specification is:

$$A(M) \triangleq \overline{c_{AB}}\langle M \rangle$$

$$B_{spec}(M) \triangleq c_{AB}(x).F(M)$$

$$Inst_{spec}(M) \triangleq (\nu c_{AB})(A(M) \mid B_{spec}(M))$$

The principal A is as usual, but the principal B is replaced with a variant $B_{spec}(M)$; this variant receives an input from A and then acts like B when B receives M. We may say that $B_{spec}(M)$ is a "magical" version of B that knows the message M sent by A, and similarly $Inst_{spec}$ is a "magical" version of $Inst$.

Although the specification and the protocol are similar in structure, the specification is more evidently "correct" than the protocol. Therefore, we take the following equivalence as our authenticity property: $Inst(M) \simeq Inst_{spec}(M)$, for all M.

In summary, we have:

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$, for all M.

Secrecy: $Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for all M and M'.

Each of these equivalences means that two processes being equated are indistinguishable, even when an active attacker is their environment. Neither of these equivalences would hold without the restriction of channel c_{AB}. We prove these equivalences in Section 6, which contains proofs for our examples.

2.3.2. *An example with channel establishment.* A more interesting variant of our first example is obtained by adding a channel establishment phase. In this phase, before they communicate any data, the principals A and B obtain a new channel with the help of a server S.

There are many different ways of establishing a channel, even at the abstract level at which we work here. The one we describe is inspired by the Wide Mouthed Frog protocol [BAN89], which has the basic structure shown in Fig. 1.

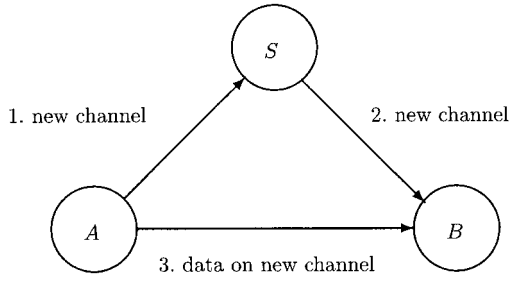

FIG. 1. Structure of the Wide Mouthed Frog protocol

We consider an abstract and simplified version of the Wide Mouthed Frog protocol. Our version is abstract in that we deal with channels instead of keys; it is simplified in that channel establishment and data communication happen only once (so there is no need for timestamps). In the next section we show how to treat keys and how to allow many instances of the protocol, with an arbitrary number of messages.

Informally, our version is:

$$\begin{array}{lll} \text{Message 1} & A \to S: c_{AB} & \text{on } c_{AS} \\ \text{Message 2} & S \to B: c_{AB} & \text{on } c_{SB} \\ \text{Message 3} & A \to B: M & \text{on } c_{AB} \end{array}$$

Here c_{AS} is a channel that A and S share initially, c_{SB} is a channel that S and B share initially, and c_{AB} is a channel that A creates for communication with B. After passing the channel c_{AB} to B through S, A sends a message M on c_{AB}. Note that S does not use the channel, but only transmits it.

In the pi calculus, we formulate this protocol as follows:

$$A(M) \triangleq (\nu c_{AB})\, \overline{c_{AS}}\langle c_{AB}\rangle.\, \overline{c_{AB}}\langle M\rangle$$

$$S \triangleq c_{AS}(x).\, \overline{c_{SB}}\langle x\rangle$$

$$B \triangleq c_{SB}(x).\, x(y).\, F(y)$$

$$Inst(M) \triangleq (\nu c_{AS})(\nu c_{SB})(A(M) \mid S \mid B)$$

Here we write $F(y)$ to represent what B does with the message y that it receives, as in the previous example. The restrictions on the channels c_{AS}, c_{SB}, and c_{AB} reflect the expected privacy guarantees for these channels. The most salient new feature of this specification is the use of scope extrusion: A generates a fresh channel c_{AB}, and then sends it out of scope to B via S. We could not have written this description in formalisms such as CCS or CSP; the use of the pi calculus is important.

For discussing authenticity, we introduce the specification:

$$A(M) \triangleq (vc_{AB})\,\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle$$

$$S \triangleq c_{AS}(x).\overline{c_{SB}}\langle x\rangle$$

$$B_{spec}(M) \triangleq c_{SB}(x).x(y).F(M)$$

$$Inst_{spec}(M) \triangleq (vc_{AS})(vc_{SB})(A(M) \mid S \mid B_{spec}(M))$$

According to this specification, the message M is communicated "magically": the process F is applied to the message M that A sends independently of whatever happens during the rest of the protocol run.

We obtain the following authenticity and secrecy properties:

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$, for all M.

Secrecy: $Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for all M and M'.

Again, these properties hold because of the scoping rules of the pi calculus.

2.3.3. *Discussion.* We believe that the two examples just given are rather encouraging. They indicate that the pi calculus is a natural language for describing some security protocols. In particular, the restriction operator and scope extrusion allow convenient representations for the possession and communication of channels.

We do not wish to suggest that the pi calculus enables us to describe all security protocols, even at an abstract level. For example, some protocols rely on asymmetric channels (channels of the kind implemented with public-key cryptography [DH76, RSA78]). It may be possible to represent such asymmetric channels in the pi calculus but extending the pi calculus may be simpler and more effective. However, the restriction operator and scope extrusion should be useful for describing security protocols even in extensions of the pi calculus.

3. PROTOCOLS USING CRYPTOGRAPHY

Just as there are several versions of the pi calculus, there are several versions of the spi calculus. These differ in particular in what cryptographic constructs they include.

In this section we introduce a relatively simple spi calculus, namely the pi calculus extended with primitives for shared-key cryptography. We then write several protocols that use shared-key cryptography in this calculus.

As in Section 2, the presentation is rather informal. Later sections contain further formal definitions. Throughout the paper, we often refer to the calculus presented in this section as "the" spi calculus; but we define other versions of the spi calculus in Section 7.

3.1. *The Spi Calculus with Shared-Key Cryptography*

The syntax of the spi calculus is an extension of that of the pi calculus. In order to represent encrypted messages, we add a clause to the syntax of terms:

$L, M, N ::=$	terms
$\cdots$	as in Section 2.2
$\{M\}_N$	shared-key encryption

In order to represent decryption, we add a clause to the syntax of processes:

$P, Q ::=$	processes
$\cdots$	as in Section 2.2
case L *of* $\{x\}_N$ *in* P	shared-key decryption

The variable x is bound in P.

Intuitively, the meaning of the new constructs is as follows:

- The term $\{M\}_N$ represents the ciphertext obtained by encrypting the term M under the key N using a shared-key cryptosystem such as DES [DES77].
- The process *case* L *of* $\{x\}_N$ *in* P attempts to decrypt the term L with the key N. If L is a ciphertext of the form $\{M\}_N$, then the process behaves as $P[M/x]$. Otherwise, the process is stuck.

Implicit in this definition are some standard but significant assumptions about cryptography:

- The only way to decrypt an encrypted packet is to know the corresponding key.
- An encrypted packet does not reveal the key that was used to encrypt it.
- There is sufficient redundancy in messages so that the decryption algorithm can detect whether a ciphertext was encrypted with the expected key.

It is not assumed that all messages contain information that allows each principal to recognize its own messages (cf. [BAN89]).

The semantics of the spi calculus can be formalized in much the same way as the semantics of the pi calculus. We carry out this formalization in Section 4. The most interesting issues in this formalization concern the notion of equivalence. Again, we write $P \simeq Q$ to mean that the behaviours of the processes P and Q are indistinguishable. However, the notion of indistinguishability is complicated by the presence of cryptography.

As an example of these complications, consider the following process:

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

This process simply sends M under a new key K on a public channel c; the key K is not transmitted. Intuitively, we would like to be able to say that $P(M)$ and

$P(M')$ are indistinguishable, for any M and M', because an observer cannot discover K and hence cannot tell whether M or M' is sent under K. On the other hand, $P(M)$ and $P(M')$ are clearly different, since they transmit different messages on c. A fine-grained equivalence—such as the standard strong bisimilarity—would distinguish $P(M)$ and $P(M')$. Our equivalence is coarse-grained enough not to make this unwanted distinction.

3.2. Examples Using Shared-Key Cryptography

The spi calculus enables more detailed descriptions of security protocols than the pi calculus. While the pi calculus enables the representation of channels, the spi calculus also enables the representation of the channel implementations in terms of cryptography. In this section we show a few example cryptographic protocols.

As in the pi calculus, scoping is the basis of security in the spi calculus. In particular, restriction can be used to model the creation of fresh, unguessable cryptographic keys. Restriction can also be used to model the creation of fresh nonces of the sort used in challenge-response exchanges.

Security properties can still be expressed as equivalences, although the notion of equivalence is more delicate, as we have discussed.

3.2.1. *A first cryptographic example.* Our first example is a cryptographic version of the example of Section 2.3.1. We consider two principals A and B that share a key K_{AB}; in addition, we assume there is a public channel c_{AB} that A and B can use for communication, but which is in no way secure. The protocol is simply that A sends a message M under K_{AB} to B, on c_{AB}.

Informally, we write this protocol as follows:

$$\text{Message 1} \quad A \rightarrow B: \{M\}_{K_{AB}} \qquad \text{on } c_{AB}$$

In the spi calculus, we write:

$$A(M) \triangleq \overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle$$

$$B \triangleq c_{AB}(x).\mathit{case}\ x\ \mathit{of}\ \{y\}_{K_{AB}}\ \mathit{in}\ F(y)$$

$$\mathit{Inst}(M) \triangleq (\nu K_{AB})(A(M) \mid B)$$

According to this definition, A sends $\{M\}_{K_{AB}}$ on c_{AB} while B listens for a message on c_{AB}. Given such a message, B attempts to decrypt it using K_{AB}; if this decryption succeeds, B applies F to the result. The assumption that A and B share K_{AB} gives rise to the restriction on K_{AB}, which is syntactically legal and meaningful although K_{AB} is not used as a channel. On the other hand, c_{AB} is not restricted, since it is a public channel. Other principals may send messages on c_{AB}, so B may attempt to decrypt a message not encrypted under K_{AB}; in that case, the protocol will get stuck. We are not concerned about this possibility, but it would be easy enough to avoid it by writing a slightly more elaborate program for B.

We use the following specification:

$$A(M) \triangleq \overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle$$

$$B_{spec}(M) \triangleq c_{AB}(x).case\ x\ of\ \{y\}_{K_{AB}}\ in\ F(M)$$

$$Inst_{spec}(M) \triangleq (\nu K_{AB})(A(M) \mid B_{spec}(M))$$

and we obtain the properties:

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$, for all M.

Secrecy: $Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for all M and M'.

Intuitively, authenticity holds even if the key K_{AB} is somehow compromised after its use. Many factors can contribute to key compromise, for example incompetence on the part of protocol participants, and malice and brute force on the part of attackers. We cannot model all these factors, but we can model deliberate key publication, which is in a sense the most extreme of them. It suffices to make a small change in the definitions of B and B_{spec}, so that they send K_{AB} on a public channel after receiving $\{M\}_{K_{AB}}$. This change preserves the authenticity equation, but clearly not the secrecy equation.

There is an apparent correspondence between the protocol of this section and that of Section 2.3.1, which does not use cryptography. Informally, we may say that this is a cryptographic implementation of the protocol of Section 2.3.1. More precisely, we conjecture that this protocol is an implementation of the parallel composition of the protocol of Section 2.3.1 with $(\nu n)\,\overline{c_{AB}}\langle n \rangle \mid c_{AB}(x).\mathbf{0}$. (Our notion of implementation is a testing preorder; see Section 4.) The role of $(\nu n)\,\overline{c_{AB}}\langle n \rangle$ is to send a decoy message on c_{AB}; similarly, the role of $c_{AB}(x).\mathbf{0}$ is to absorb a message on c_{AB}. These processes are needed because an environment can detect whether c_{AB} is used or not, and hence (in absence of these processes) can distinguish the protocol of this section from that of Section 2.3.1.

We do not study implementation relations in this paper. However, we do believe that such relations are important and that they deserve more attention in the field of security. We view this example of an implementation relation as an intriguing novelty; it suggests the possibility of hierarchical development of cryptographic protocols from non-cryptographic specifications.

3.2.2. *An example with key establishment.* In cryptographic protocols, the establishment of new channels often means the exchange of new keys. There are many methods (most of them flawed) for key exchange. The following example is the cryptographic version of that of Section 2.3.2, and uses a simplified form of the Wide Mouthed Frog key exchange. The example is represented in Fig. 2.

In the Wide Mouthed Frog protocol, the principals A and B share keys K_{AS} and K_{SB} respectively with a server S. When A and B want to communicate securely, A creates a new key K_{AB}, sends it to the server under K_{AS}, and the server forwards it to B under K_{SB}. Since all communication is protected by encryption, communication

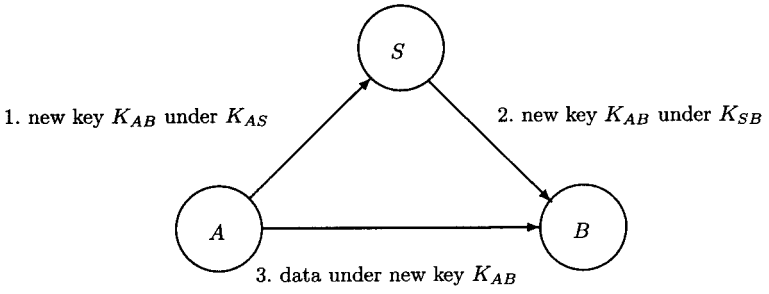

FIG. 2. Sketch of the Wide Mouthed Frog protocol.

can take place through public channels, which we write c_{AS}, c_{SB}, and c_{AB}. Informally, a simplified version of this protocol is:

$$\begin{array}{lll} \text{Message 1} & A \to S: \{K_{AB}\}_{K_{AS}} & \text{on } c_{AS} \\ \text{Message 2} & S \to B: \{K_{AB}\}_{K_{SB}} & \text{on } c_{SB} \\ \text{Message 3} & A \to B: \{M\}_{K_{AB}} & \text{on } c_{AB} \end{array}$$

In the spi calculus, we can express this message sequence as follows:

$$\begin{aligned} A(M) &\triangleq (\nu K_{AB})(\overline{c_{AS}}\langle \{K_{AB}\}_{K_{AS}} \rangle . \overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle) \\ S &\triangleq c_{AS}(x).\mathit{case}\ x\ \mathit{of}\ \{y\}_{K_{AS}}\ \mathit{in}\ \overline{c_{SB}}\langle \{y\}_{K_{SB}} \rangle \\ B &\triangleq c_{SB}(x).\mathit{case}\ x\ \mathit{of}\ \{y\}_{K_{SB}}\ \mathit{in} \\ &\qquad c_{AB}(z).\mathit{case}\ z\ \mathit{of}\ \{w\}_{y}\ \mathit{in}\ F(w) \\ \mathit{Inst}(M) &\triangleq (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B) \end{aligned}$$

where $F(w)$ is a process representing the rest of the behaviour of B upon receiving a message w. Notice the essential use of scope extrusion: A generates the key K_{AB} and sends it out of scope to B via S.

Following our usual pattern, we introduce a specification for discussing authenticity:

$$\begin{aligned} A(M) &\triangleq (\nu K_{AB})(\overline{c_{AS}}\langle \{K_{AB}\}_{K_{AS}} \rangle . \overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle) \\ S &\triangleq c_{AS}(x).\mathit{case}\ x\ \mathit{of}\ \{y\}_{K_{AS}}\ \mathit{in}\ \overline{c_{SB}}\langle \{y\}_{K_{SB}} \rangle \\ B_{spec}(M) &\triangleq c_{SB}(x).\mathit{case}\ x\ \mathit{of}\ \{y\}_{K_{SB}}\ \mathit{in} \\ &\qquad c_{AB}(z).\mathit{case}\ z\ \mathit{of}\ \{w\}_{y}\ \mathit{in}\ F(M) \\ \mathit{Inst}_{spec}(M) &\triangleq (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B_{spec}(M)) \end{aligned}$$

One may be concerned about the apparent complexity of this specification. On the other hand, despite its complexity, the specification is still more evidently "correct" than the protocol. In particular, it is still evident that $B_{spec}(M)$ applies F

to the data M from A, rather than to some other message chosen as the result of error or attack.

We obtain the usual properties of authenticity and secrecy:

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$, for all M.

Secrecy: $Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for all M and M'.

3.2.3. *A complete authentication example (with a flaw).* In the examples discussed so far, channel establishment and data communication happen only once. As we demonstrate now, it is a simple matter of programming to remove this restriction and to represent more sophisticated examples with many sessions between many principals. However, as the intricacy of our examples increases, so does the opportunity for error. This should not be construed as a limitation of our approach, but rather as the sign of an intrinsic difficulty: many of the mistakes in authentication protocols arise from confusion between sessions.

We consider a system with a server S and n other principals. We use the terms $suc(0)$, $suc(suc(0))$, ..., which we abbreviate to $\underline{1}, \underline{2}, ...$, as the names of these other principals. We assume that each principal has an input channel; these input channels are public and have the names c_1, c_2, ..., c_n and c_S. We also assume that the server shares a pair of keys with each other principal, one key for each direction: principal i uses a key K_{iS} to send to S and a different key K_{Si} to receive from S, for $1 \leqslant i \leqslant n$. In the Wide Mouthed Frog protocol, as in many other small protocols, the keys K_{iS} and K_{Si} are identical; our use of two different keys simplifies reasoning by making it impossible to confuse certain messages.

We extend our standard example to this system of $n+1$ principals, with the following message sequence:

Message 1	$A \to S$: $A, \{B, K_{AB}\}_{K_{AS}}$	on c_S
Message 2	$S \to B$: $\{A, K_{AB}\}_{K_{SB}}$	on c_B
Message 3	$A \to B$: $A, \{M\}_{K_{AB}}$	on c_B

Here A and B range over the n principals. The names A and B appear in messages in order to avoid ambiguity; when these names appear in clear, they function as hints that help the recipient choose the appropriate key for decryption of the rest of the message. The intent is that the protocol can be used by any pair of principals, arbitrarily often; concurrent runs are allowed.

As it stands, the protocol is seriously flawed; a correct protocol appears below, in Section 3.2.4. (The flaws and their fixes should be clear to readers knowledgeable in security.) However, we continue to discuss the protocol in order to explain our method for representing it in the spi calculus.

In our spi calculus representation, we use several convenient abbreviations. First, we rely on pair splitting on input and on decryption:

$$c(x_1, x_2).P \triangleq c(y).let\ (x_1, x_2) = y\ in\ P$$

$$case\ L\ of\ \{x_1, x_2\}_N\ in\ P \triangleq case\ L\ of\ \{y\}_N\ in\ let\ (x_1, x_2) = y\ in\ P$$

where variable y does not occur free in P. Second, we need the standard notation for the composition of a finite set of processes. Given a finite family of processes $P_1, ..., P_k$, we let $\prod_{i \in 1..k} P_i$ be their k-way composition $P_1 \mid \cdots \mid P_k$. Finally, we omit the inner brackets from an encrypted pair of the form $\{(N, N')\}_{N''}$, and simply write $\{N, N'\}_{N''}$, as is common in informal descriptions.

Informally, an instance of the protocol is determined by a choice of parties (who is A and who is B) and by the message sent after key establishment. More formally, an instance I is a triple (i, j, M) such that i and j are principals and M is a message. We say that i is the source address and j the destination address of the instance. Moreover, we assume that there is an abstraction F representing the behaviour of any principal after receipt of Message 3 of the protocol. For an instance (i, j, M) that runs as intended, the argument to F is the triple $(\underline{i}, \underline{j}, M)$.

Given an instance (i, j, M), the following process corresponds to the role of A:

$$Send(i, j, M) \triangleq (\nu K)(\overline{c_S}\langle(\underline{i}, \{\underline{j}, K\}_{K_{iS}})\rangle \mid \overline{c_j}\langle(\underline{i}, \{M\}_K)\rangle).$$

The sending process creates a key K and sends it to the server, along with the names $\underline{i}$ and $\underline{j}$ of the principals of the instance. The sending process also sends M under K, along with its name $\underline{i}$. We have put the two messages in parallel, somewhat arbitrarily; but putting them in sequence would have much the same effect.

The following process corresponds to the role of B for principal j:

$$\begin{aligned} Recv(j) \triangleq\ & c_j(y_{\text{cipher}}).\, case\ y_{cipher}\ of\ \{x_A, x_{key}\}_{K_{Sj}}\ in \\ & c_j(z_A, z_{cipher}).\, [x_A\ is\ z_A] \\ & case\ z_{cipher}\ of\ \{z_{plain}\}_{x_{key}}\ in\ F(x_A, \underline{j}, z_{plain}) \end{aligned}$$

The receiving process waits for a message y_{cipher} from the server, extracts a key x_{key} from this message, then waits for a message z_{cipher} under this key, and finally applies F to the name x_A of the presumed sender, to its own name $\underline{j}$, and to the contents z_{plain} of the message. The variables x_A and z_A are both intended as the name of the sending process, so they are expected to match.

The server S is the same for all instances:

$$\begin{aligned} S \triangleq\ & c_S(x_A, x_{cipher}). \\ & \prod_{i \in 1..n} [x_A\ is\ \underline{i}]\ case\ x_{cipher}\ of\ \{x_B, x_{key}\}_{K_{iS}}\ in \\ & \prod_{j \in 1..n} [x_B\ is\ \underline{j}]\ \overline{c_j}\langle\{x_A, x_{key}\}_{K_{Sj}}\rangle \end{aligned}$$

The variable x_A is intended as the name of the sending process, x_B as the name of the receiving process, x_{key} as the new key, and x_{cipher} as the encrypted part of the first message of the protocol. In the code for the server, we program an n-way branch on the name x_A by using a parallel composition of processes indexed by $i \in 1..n$. We also program an n-way branch on the name x_B, similarly. (This casual

use of multiple threads is characteristic of the pi calculus; in practice the branch could be implemented more efficiently, but here we are interested only in the behaviour of the server, not in its efficient implementation.)

Finally, we define a whole system, parameterized on a list of instances of the protocol:

$$\begin{aligned} Sys(I_1, ..., I_m) \triangleq\ & (\nu \overrightarrow{K_{iS}})(\nu \overrightarrow{K_{Sj}}) \\ & (Send(I_1) \mid \cdots \mid Send(I_m) \mid \\ & !\, S \mid \\ & !\, Recv(1) \mid \cdots \mid !\, Recv(n)) \end{aligned}$$

where $(\nu \overrightarrow{K_{iS}})(\nu \overrightarrow{K_{Sj}})$ stands for $(\nu K_{1S})\ldots(\nu K_{nS})(\nu K_{S1})\ldots(\nu K_{Sn})$. The expression $Sys(I_1, ..., I_m)$ represents a system with m instances of the protocol. The server is replicated; in addition, the replication of the receiving processes means that each principal is willing to play the role of receiver in any number of runs of the protocol in parallel. Thus, any two runs of the protocol can be simultaneous, even if they involve the same principals.

As before, we write a specification by modifying the protocol. The style of this specification is somewhat more complex than that used in previous examples, but it has the advantage of accommodating multiple sessions. For this specification, we revise both the sending process and the receiving process, but not the server:

$$\begin{aligned} Send_{spec}(i, j, M) \triangleq\ & (\nu p)(Send(i, j, p) \mid p(x).F(\underline{i}, \underline{j}, M)) \\ Recv_{spec}(j) \triangleq\ & c_j(y_{cipher}).\, case\ y_{cipher}\ of\ \{x_A, x_{key}\}_{K_{Sj}}\ in \\ & c_j(z_A, z_{cipher}).[x_A\ is\ z_A] \\ & case\ z_{cipher}\ of\ \{z_{plain}\}_{x_{key}}\ in\ \overline{z_{plain}}\langle * \rangle \\ Sys_{spec}(I_1, ..., I_m) \triangleq\ & (\nu \overrightarrow{K_{iS}})(\nu \overrightarrow{K_{Sj}}) \\ & (Send_{spec}(I_1) \mid \cdots \mid Send_{spec}(I_m) \mid \\ & !\, S \mid \\ & !\, Recv_{spec}(1) \mid \cdots \mid !\, Recv_{spec}(n)) \end{aligned}$$

In this specification, the sending process for instance (i, j, M) is as in the implementation, except that it sends a fresh channel name p instead of M, and runs $F(\underline{i}, \underline{j}, M)$ when it receives any message on p. The receiving process in the specification is identical to that in the implementation, except that $F(y_A, \underline{j}, z_{plain})$ is replaced with $\overline{z_{plain}}\langle * \rangle$, where the symbol * represents a fixed but arbitrary message. The variable z_{plain} will be bound to the fresh name p for the corresponding instance of the protocol. Thus, the receiving process will signal on p, triggering the execution of the appropriate process $F(\underline{i}, \underline{j}, M)$.

A crucial property of this specification is that the only occurrences of F are bundled into the description of the sending process. There, F is applied to the desired

parameters, $(\underline{i}, \underline{j}, M)$. Hence it is obvious that an instance (i, j, M) will cause the execution of $F(\underline{i}', \underline{j}', M')$ only if i' is i, j' is j, and M' is M. Therefore, despite its complexity, the specification is more obviously "correct" than the implementation.

Much as in previous examples, we would like the protocol to have the following authenticity property:

$$Sys(I_1, ..., I_m) \simeq Sys_{spec}(I_1, ..., I_m), \qquad \text{for any instances } I_1, ..., I_m.$$

Unfortunately, the protocol is vulnerable to a replay attack that invalidates the authenticity equation. Consider the system $Sys(I, I')$ where $I = (i, j, M)$ and $I' = (i, j, M')$. An attacker can replay messages of one instance and get them mistaken for messages of the other instance, causing M to be passed twice to F. Thus, $Sys(I, I')$ can be made to execute two copies of $F(\underline{i}, \underline{j}, M)$. In contrast, no matter what an attacker does, $Sys_{spec}(I, I')$ will run each of $F(\underline{i}, \underline{j}, M)$ and $F(\underline{i}, \underline{j}, M')$ at most once. The authenticity equation therefore does not hold. We disprove it more formally in Section 6.3.

We leave the discussion of secrecy for the next example.

3.2.4. *A complete authentication example* (*repaired*). We now improve the protocol of the previous section by adding nonce handshakes as protection against replay attacks. The Wide Mouthed Frog protocol uses timestamps instead of handshakes. The treatment of timestamps in the spi calculus is possible, but it requires additional elements, including at least a rudimentary account of clock synchronization. Protocols that use handshakes are fundamentally more self-contained than protocols that use timestamps; therefore, handshakes make for clearer examples.

Informally, our new protocol is:

Message 1	$A \to S: A$	on c_S
Message 2	$S \to A: N_S$	on c_A
Message 3	$A \to S: A, \{A, A, B, K_{AB}, N_S\}_{K_{AS}}$	on c_S
Message 4	$S \to B: *$	on c_B
Message 5	$B \to S: N_B$	on c_S
Message 6	$S \to B: \{S, A, B, K_{AB}, N_B\}_{K_{SB}}$	on c_B
Message 7	$A \to B: A, \{M\}_{K_{AB}}$	on c_B

Messages 1 and 2 are the request for a challenge and the challenge, respectively. The challenge is N_S, a nonce created by S; the nonce must not have been used previously for this purpose. Obviously the nonce is not secret, but it must be unpredictable (for otherwise an attacker could simulate a challenge and later replay the response [AN96]). In Message 3, A says that A and B can communicate under K_{AB}, sometime after receipt of N_S. All the components A, B, K_{AB}, N_S appear explicitly in the message, for safety [AN96] but A could perhaps be elided. The presence of N_S in Message 3 proves the freshness of the message. In Message 4, $*$ represents a fixed but arbitrary message; S uses $*$ to signal that it is ready for a

nonce challenge N_B from B. In Message 6, S says that A says that A and B can communicate under K_{AB}, sometime after receipt of N_B. The first field of the encrypted portions of Messages 3 and 6 (A or S) makes explicit the senders of the messages (somewhat redundantly). Finally, Message 7 is the transmission of data under K_{AB}.

The messages of this protocol have many components. For the spi calculus representation it is therefore convenient to generalize our syntax of pairs to arbitrary tuples. We use the following standard abbreviation, given inductively for any $k \geqslant 2$,

$$(N_1, ..., N_k, N_{k+1}) \triangleq ((N_1, ..., N_k), N_{k+1})$$

and similarly we write $let\ (x_1, ..., x_k) = N\ in\ P$, $c(x_1, ..., x_k).P$, and $case\ L\ of\ \{x_1, ..., x_k\}_N\ in\ P$.

In the spi calculus, we represent the nonces of this protocol as newly created names. We obtain the following spi calculus expressions:

$$\begin{aligned}
Send(i, j, M) \triangleq\ & \overline{c_S}\langle i\rangle \mid \\
& c_i(x_{nonce}). \\
& (\nu K)(\overline{c_S}\langle(\underline{i}, \{\underline{i}, \underline{i}, \underline{j}, K, x_{nonce}\}_{K_{iS}})\rangle \mid \overline{c_j}\langle(\underline{i}, \{M\}_K)\rangle) \\
S \triangleq\ & c_S(x_A). \prod_{i \in 1..n} [x_A\ is\ \underline{i}](\nu N_S)(\overline{c_i}\langle N_S\rangle \mid \\
& c_S(x'_A, x_{cipher}).[x'_A\ is\ \underline{i}] \\
& case\ x_{cipher}\ of\ \{y_A, z_A, x_B, x_{key}, x_{nonce}\}_{K_{iS}}\ in \\
& \prod_{j \in 1..n} [y_A\ is\ \underline{i}][z_A\ is\ \underline{i}][x_B\ is\ \underline{j}][x_{nonce}\ is\ N_S] \\
& (\overline{c_j}\langle *\rangle \mid c_S(y_{nonce}).\overline{c_j}\langle\{S, \underline{i}, \underline{j}, x_{key}, y_{nonce}\}_{K_{Sj}}\rangle)) \\
Recv(j) \triangleq\ & c_j(w).(\nu N_B)(\overline{c_S}\langle N_B\rangle \mid \\
& c_j(y_{cipher}). \\
& case\ y_{cipher}\ of\ \{x_S, x_A, x_B, x_{key}, y_{nonce}\}_{K_{Sj}}\ in \\
& \prod_{i \in 1..n} [x_S\ is\ S][x_A\ is\ \underline{i}][x_B\ is\ \underline{j}][y_{nonce}\ is\ N_B] \\
& c_j(z_A, z_{cipher}).[z_A\ is\ x_A] \\
& case\ z_{cipher}\ of\ \{z_{plain}\}_{x_{key}}\ in\ F(\underline{i}, \underline{j}, z_{plain})) \\
Sys(I_1, ..., I_m) \triangleq\ & (\nu \overrightarrow{K_{iS}})(\nu \overrightarrow{K_{Sj}}) \\
& (Send(I_1) \mid \cdots \mid Send(I_m) \mid \\
& !S \mid \\
& !Recv(1) \mid \cdots \mid !Recv(n))
\end{aligned}$$

The names N_S and N_B represent the nonces. The variable subscripts are hints that indicate what the corresponding variables should represent; for example, x_A, x'_A, y_A, and z_A are all expected to be the name of the sending process, and x_{nonce} and y_{nonce} are expected to be the nonces generated by S and B, respectively.

The definition of Sys_{spec} is exactly analogous to that of the previous section, so we omit it.

We now obtain the authenticity property:

$$Sys(I_1, ..., I_m) \simeq Sys_{spec}(I_1, ..., I_m), \qquad \text{for any instances } I_1, ..., I_m.$$

This property holds because of the use of nonces. In particular, the attack described in Section 3.2.3 can no longer distinguish $Sys(I_1, ..., I_m)$ from $Sys_{spec}(I_1, ..., I_m)$.

As a secrecy property, we would like to express that there is no way for an external observer to tell apart two executions of the system with identical participants but different messages. The secrecy property should therefore assert that the protocol does not reveal any information about the contents of exchanged messages if none is revealed after the key exchange.

In order to express that no information is revealed after the key exchange, we introduce the following definition. We say that a pair of instances (i, j, M) and (i', j', M') is indistinguishable if the two instances have the same source and destination addresses ($i = i'$ and $j = j'$) and if $F(\underline{i}, \underline{j}, M) \simeq F(\underline{i}, \underline{j}, M')$.

Our definition of secrecy is that, if each pair $(I_1, J_1), ..., (I_m, J_m)$ is indistinguishable, then $Sys(I_1, ..., I_m) \simeq Sys(J_1, ..., J_m)$. This means that an observer cannot distinguish two systems parameterized by two sets of indistinguishable instances. This property holds for our protocol.

In summary, we have:

Authenticity: $Sys(I_1, ..., I_m) \simeq Sys_{spec}(I_1, ..., I_m)$,
for any instances $I_1, ..., I_m$.

Secrecy: $Sys(I_1, ..., I_m) \simeq Sys(J_1, ..., J_m)$,
if each pair $(I_1, J_1), ..., (I_m, J_m)$ is indistinguishable.

We could ask for a further property of anonymity, namely that the source and the destination addresses of instances be protected from eavesdroppers. However, anonymity holds neither for our protocol nor for most current, practical protocols. It would be easy enough to specify anonymity, should it be relevant.

As suggested in Section 3.2.1, we could also consider a variant of the protocol where some keys are compromised. For this protocol, the compromised keys could include both session keys and longer-term keys shared with S. Allowing the longer-term keys K_{iS} and K_{Si} to be compromised is basically equivalent to considering the case where principal i may behave dishonestly and not follow the protocol. We believe that, even in the presence of dishonest principals, the protocol guarantees security for sessions between honest principals.

3.2.5. *Discussion.* After these examples, it should be obvious that writing a protocol in the spi calculus is a little harder than writing it in the informal notations common in the literature. On the other hand, the spi calculus versions are

more detailed. They make clear not only what messages are sent but also how the messages are generated and how they are checked. These aspects of the spi calculus descriptions add complexity, but they enable finer analysis. (Recall, for example, that one of the mistakes in the CCITT X.509 protocol was to omit a timestamp check [BAN89].)

It should also be obvious that writing a protocol in the spi calculus is essentially analogous to writing it in any programming language with suitable communication and encryption libraries. The main advantage of the spi calculus is its formal precision.

We cannot say that the spi calculus will be as good a tool for finding flaws in protocols as some of the logics listed in the introduction. On the other hand, the spi calculus seems to rest on firmer ground, so it yields more convincing proofs of correctness.

4. FORMAL SEMANTICS OF THE SPI CALCULUS

In this section, we start the formal treatment of the spi calculus. In Section 4.1 we introduce the reaction relation; $P \to Q$ means there is a reaction between subprocesses of P such that the whole can take a step to process Q. Reaction is the basic notion of computation in both the pi calculus and the spi calculus. In Section 4.2, we give a precise definition of the equivalence relation $\simeq$, which we have used for expressing security properties.

Syntactic Conventions

The grammar of the spi calculus is given in Sections 2.2 and 3.1. It has two syntactic categories, of *terms*, ranged over by L, M, N, and of *processes*, ranged over by P, Q, R. The metavariables m, n, p, q, and r range over an infinite set of *names*. The metavariables x, y, and z range over a disjoint, infinite set of *variables*.

We write $fn(M)$ and $fn(P)$ for the sets of names free in term M and process P respectively. Similarly, we write $fv(M)$ and $fv(P)$ for the sets of variables free in M and P, respectively. We say that a term or process is *closed* to mean that it has no free variables. (To be able to communicate externally, a process must have free names.) The set $Proc = \{P \mid fv(P) = \varnothing\}$ is the set of closed processes.

4.1. The Reaction Relation

The reaction relation is a concise account of computation in the pi calculus introduced by Milner [Mil92], inspired by the Chemical Abstract Machine of Berry and Boudol [BB92]. One thinks of a process as consisting of a chemical solution of molecules waiting to react. A reaction step arises from the interaction of the adjacent molecules $\bar{m}\langle N\rangle.P$ and $m(x).Q$, as follows:

$$\text{(React Inter)} \quad \bar{m}\langle N\rangle.P \mid m(x).Q \to P \mid Q[N/x]$$

Just as one might stir a chemical solution to allow non-adjacent molecules to react, we define a relation, *structural equivalence*, that allows processes to be rearranged so that (React Inter) is applicable. We first define the *reduction relation* $>$ on closed processes:

(Red Repl) $\quad !P > P \mid !P$

(Red Match) $\quad [M \text{ is } M]\, P > P$

(Red Let) $\quad let(x, y) = (M, N) \text{ in } P > P[M/x][N/y]$

(Red Zero) $\quad case\ 0\ of\ 0: P\ suc(x): Q > P$

(Red Suc) $\quad case\ suc(M)\ of\ 0: P\ suc(x): Q > Q[M/x]$

(Red Decrypt) $\quad case\ \{M\}_N\ of\ \{x\}_N\ in\ P > P[M/x]$

(The reduction relation is not found in previous accounts of the pi calculus; we introduce it here because it is useful also in the definition of commitment, given in Section 5.1.) We let structural equivalence, $\equiv$, be the least relation on closed processes that satisfies the following equations and rules:

(Struct Nil) $\quad P \mid \mathbf{0} \equiv P$

(Struct Comm) $\quad P \mid Q \equiv Q \mid P$

(Struct Assoc) $\quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

(Struct Switch) $\quad (vm)(vn)\, P \equiv (vn)(vm)\, P$

(Struct Drop) $\quad (vn)\, \mathbf{0} \equiv \mathbf{0}$

(Struct Extrusion) $\quad (vn)(P \mid Q) \equiv P \mid (vn)\, Q \qquad \text{if} \quad n \notin fn(P)$

(Struct Red) $\quad \dfrac{P > Q}{P \equiv Q}$

(Struct Refl) $\quad \dfrac{}{P \equiv P}$

(Struct Symm) $\quad \dfrac{P \equiv Q}{Q \equiv P}$

(Struct Trans) $\quad \dfrac{P \equiv Q \quad Q \equiv R}{P \equiv R}$

(Struct Par) $\quad \dfrac{P \equiv P'}{P \mid Q \equiv P' \mid Q}$

(Struct Res) $\quad \dfrac{P \equiv P'}{(vm)\, P \equiv (vm)\, P'}$

Now we can complete the formal description of the reaction relation. We let the reaction relation, $\rightarrow$, be the least relation on closed processes that satisfies (React Inter) and the following rules:

$$\text{(React Struct)}\quad \frac{P \equiv P' \quad P' \to Q' \quad Q' \equiv Q}{P \to Q}$$

$$\text{(React Par)}\quad \frac{P \to P'}{P \mid Q \to P' \mid Q} \qquad \text{(React Res)}\quad \frac{P \to P'}{(\nu n)\, P \to (\nu n)\, P'}.$$

This definition of the reaction relation corresponds to the informal description of process behaviour given in Sections 2.2 and 3.1.

As an example, we can use the definition of the reaction relation to show the behaviour of the protocol of Section 3.2.2.

$$\begin{aligned}
Inst(M) &\equiv (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B) \\
&\to (\nu K_{AS})(\nu K_{SB})(\nu K_{AB}) \\
&\qquad (\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid \overline{c_{SB}}\langle \{K_{AB}\}_{K_{SB}} \rangle \mid B) \\
&\to (\nu K_{AS})(\nu K_{SB})(\nu K_{AB}) \\
&\qquad (\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid c_{AB}(z).\mathit{case}\ z\ \mathit{of}\ \{w\}_{K_{AB}}\ \mathit{in}\ F(w)) \\
&\to (\nu K_{AS})(\nu K_{SB})(\nu K_{AB})\, F(M) \\
&\equiv F(M)
\end{aligned}$$

The last step in this calculation is justified by our general convention that none of the bound parameters of the protocol (including, in this case, K_{AS}, K_{SB}, and K_{AB}) occurs free in F.

4.2. *Testing Equivalance*

In order to define testing equivalence, we first define a predicate that describes the channels on which a process can communicate. We let a *barb*, β, be an input or output channel, that is, either a name m (representing input) or a *co-name* $\bar{m}$ (representing output). For a closed process P, we define the predicate P *exhibits barb* β, written $P \downarrow \beta$, by the two axioms:

$$\text{(Barb In)}\quad m(x).P \downarrow m \qquad\qquad \text{(Barb Out)}\quad \bar{m}\langle M \rangle.P \downarrow \bar{m}$$

and the three rules

$$\text{(Barb Par)}\quad \frac{P \downarrow \beta}{P \mid Q \downarrow \beta} \qquad \text{(Barb Res)}\quad \frac{P \downarrow \beta \quad \beta \notin \{m, \bar{m}\}}{(\nu m)\, P \downarrow \beta} \qquad \text{(Barb Struct)}\quad \frac{P \equiv Q \quad Q \downarrow \beta}{P \downarrow \beta}$$

Intuitively, $P \downarrow \beta$ holds just if P is a closed process that may input or output immediately on barb β. The convergence predicate $P \Downarrow \beta$ holds if P is a closed process that exhibits β after some reactions:

$$\text{(Conv Barb)} \qquad \frac{P \downarrow \beta}{P \Downarrow \beta} \qquad\qquad \text{(Conv React)} \qquad \frac{P \to Q \quad Q \Downarrow \beta}{P \Downarrow \beta}$$

We let a *test* consist of any closed process R and any barb β. A closed process P passes the test if and only if $(P \mid R) \Downarrow \beta$. The notion of testing gives rise to a testing preorder $\sqsubseteq$ and to a testing equivalence $\simeq$ on the set *Proc* of closed processes:

$$P \sqsubseteq Q \triangleq \text{for any test } (R, \beta), \text{ if } (P \mid R) \Downarrow \beta \text{ then } (Q \mid R) \Downarrow \beta$$

$$P \simeq Q \triangleq P \sqsubseteq Q \text{ and } Q \sqsubseteq P$$

The idea of testing equivalence comes from the work of De Nicola and Hennessy [DH84]. In that work, tests are processes that contain the distinguished name ω (instead of being parameterized by a barb β). This is only a superficial difference, and we can show that our relation $\simeq$ is a version of De Nicola and Hennessy's may-testing equivalence. As De Nicola and Hennessy have explained, may-testing corresponds to partial correctness (or safety), while must-testing corresponds to total correctness. Like much of the security literature, our work focuses on safety properties, hence our definitions.

One of the advantages of testing equivalence as the basis of our specifications of authenticity and secrecy is its simple definition in terms of the convergence predicate. A test neatly formalizes the idea of a generic experiment or observation that another process (such as an attacker) might perform on a process. Thus testing equivalence concisely captures the concept of equivalence in an arbitrary environment.

According to our definitions, two closed processes P and Q are testing equivalent if their respective parallel compositions with a third process R behave similarly. It follows that P and Q can be used interchangeably in any context (not just in parallel with R). More precisely, testing equivalence is a congruence; that is, $\simeq$ is an equivalence relation with the property that if $P \simeq Q$ then $\mathscr{C}[P] \simeq \mathscr{C}[Q]$ for any closed context $\mathscr{C}$. (A closed context $\mathscr{C}$ is a closed process with a single hole; $\mathscr{C}[P]$ and $\mathscr{C}[Q]$ are the outcomes of filling the hole with P and Q, respectively.)

PROPOSITION 1. (1) *Structural equivalence implies testing equivalence.*

(2) *Testing equivalence is reflexive, transitive, and symmetric.*

(3) *Testing equivalence is a congruence on closed processes.*

This proposition is essential for equational reasoning with testing equivalence. Its proof is in Appendix C, where we show that testing equivalence remains a congruence when extended to open processes.

Testing equivalence is sensitive to the choice of language. Two processes that are testing equivalent in our calculus may not be testing equivalent after new constructs are added to the calculus. As Boreale and De Nicola have shown [BN95], testing equivalence becomes finer-grained in the presence of a mismatch construct ([M *is not* N] P). Our calculus does not include a mismatch construct because we have not found a need for it in writing protocols; however, such a construct is sensible and perhaps yields a better definition of testing equivalence. The same is true for other "negative" constructs that check whether a term is not a name, not a number, not a pair, or not encrypted under a given key. We believe that the results of this paper remain valid for a range of reasonable extensions of our calculus, but we leave the study of such extensions for future work.

5. SEMANTIC NOTIONS USEFUL IN PROOFS

This section develops proof techniques for the spi calculus, based on earlier work on the pi calculus. Section 5.1 defines the *commitment relation*, providing in particular a characterization of the reaction relation. Section 5.2 reviews the notions of *strong bisimulation*, *barbed equivalence*, and *barbed congruence* [MS92]. Finally, Section 5.3 introduces the *underpinning relation* and shows its use for proofs of secrecy.

In order to prove a testing equivalence directly, we need to consider arbitrary tests and arbitrary sequences of reactions. The use of structural equivalence to define reaction is elegant, but makes proofs a little awkward. One of the purposes of this section is to obtain a direct inductive characterization of reaction without appeal to structural equivalence, and a co-inductive method for proving testing equivalence.

5.1. The Commitment Relation

The original semantics of the pi calculus (given in [MPW92]) is not based on the notion of reaction, but rather on a labelled transition system. Here we define a labelled-transition semantics for the spi calculus, imitating Milner's recent lecture notes [Mil95b]. Despite differences in style, this semantics is essentially equivalent to the one of Section 4, so it can be used in proofs about that semantics.

We need some new syntactic forms: abstractions, concretions, and agents. An *abstraction* is an expression of the form $(x)\,P$, where x is a bound variable and P is a process. Intuitively, $(x)\,P$ is like the process $p(x).P$ minus the name p. A *concretion* is an expression of the form $(\nu m_1, ..., m_k)\langle M\rangle\,P$, where M is a term, P is a process, $k \geqslant 0$, and the names m_1, ..., m_k are bound in M and P. Intuitively, $(\nu m_1, ..., m_k)\langle M\rangle\,P$ is like the process $(\nu m_1)\ldots(\nu m_k)\,\bar{p}\langle M\rangle\,P$ minus the name p, provided p is not one of m_1, ..., m_k. We often write concretions as $(\nu \vec{m})\langle M\rangle\,P$, where $\vec{m} = m_1, ..., m_k$, or simply $(\nu)\langle M\rangle\,P$ if $k = 0$. Finally, an agent is an abstraction, a process, or a concretion. We use the metavariables A and B to stand for arbitrary agents, and let $fv(A)$ and $fn(A)$ be the sets of free variables and names of an agent A, respectively.

We extend the restriction and composition operators to arbitrary agents, as follows. For an abstraction, $(x)\,P$, we set:

$$(vm)(x)\,P \triangleq (x)(vm)\,P$$
$$R \mid (x)\,P \triangleq (x)(R \mid P)$$

assuming that $x \notin fv(R)$. For a concretion, $(v\vec{n})\langle M\rangle\,Q$, we set:

$$(vm)(v\vec{n})\langle M\rangle\,Q \triangleq \begin{cases}(vm, \vec{n})\langle M\rangle\,Q & \text{if } m \in fn(M)\\ (v\vec{n})\langle M\rangle(vm)\,Q & \text{otherwise}\end{cases}$$
$$R \mid (v\vec{n})\langle M\rangle\,Q \triangleq (v\vec{n})\langle M\rangle(R \mid Q)$$

assuming that $m \notin \{\vec{n}\}$ and that $\{\vec{n}\} \cap fn(R) = \varnothing$. We define the dual composition $A \mid R$ symmetrically. If F is the abstraction $(x)\,P$ and C is the concretion $(v\vec{n})\langle M\rangle\,Q$, and $\{\vec{n}\} \cap fn(P) = \varnothing$, we define the *interactions* $F@C$ and $C@F$ to be the processes given by:

$$F@C \triangleq (v\vec{n})(P[M/x] \mid Q)$$
$$C@F \triangleq (v\vec{n})(Q \mid P[M/x])$$

When F is the abstraction $(x)\,P$, we may write $F(M)$ for its instantiation to M, that is, for $P[M/x]$. With this notation, we have $F@C = (v\vec{n})(F(M) \mid Q)$ and $C@F = (v\vec{n})(Q \mid F(M))$. Intuitively, these processes are the possible immediate results of the encounter of F and C. Given a common name p, we have that F is like $p(x).P$ and C is like $(v\vec{n})\,\bar{p}\langle M\rangle\,P$, so an interaction of F and C is a process obtained when $p(x).P$ and $(v\vec{n})\,\bar{p}\langle M\rangle\,P$, put in parallel, communicate on p.

An *action* is a name m, a co-name $\bar{m}$, or the distinguished *silent action* τ. That is, an action is either a barb or τ. The *commitment relation* is written $P \xrightarrow{\alpha} A$, where P is a closed process, α is an action, and A is a closed agent. We define this relation inductively, by the following rules:

(Comm In)
$$\overline{m(x).P \xrightarrow{m} (x)\,P}$$

(Comm Out)
$$\overline{\bar{m}\langle M\rangle.P \xrightarrow{\bar{m}} (v)\langle M\rangle\,P}$$

(Comm Inter 1)
$$\frac{P \xrightarrow{m} F \quad Q \xrightarrow{\bar{m}} C}{P \mid Q \xrightarrow{\tau} F@C}$$

(Comm Inter 2)
$$\frac{P \xrightarrow{\bar{m}} C \quad Q \xrightarrow{m} F}{P \mid Q \xrightarrow{\tau} C@F}$$

(Comm Par 1)
$$\frac{P \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} A \mid Q}$$

(Comm Par 2)
$$\frac{Q \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} P \mid A}$$

(Comm Res)
$$\frac{P \xrightarrow{\alpha} A \quad \alpha \notin \{m, \bar{m}\}}{(vm)\,P \xrightarrow{\alpha} (vm)\,A}$$

(Comm Red)
$$\frac{P > Q \quad Q \xrightarrow{\alpha} A}{P \xrightarrow{\alpha} A}$$

Intuitively, (Comm In) says that an abstraction is the residue of an input commitment; (Comm Out) says that a concretion is the residue of an output commitment; and (Comm Inter 1) and (Comm Inter 2) say that the combination of an abstraction and a concretion gives an interaction. Thus, the commitment relation has a straightforward structural definition; that is its main appeal.

Whenever $P \xrightarrow{\alpha} A$, the action α is τ, a name, or a co-name just if the agent A is a process, an abstraction, or a concretion, respectively. Therefore, the commitment relation indexed by τ, $\xrightarrow{\tau}$, is a binary relation on closed processes. We write $\xrightarrow{\tau}{}^*$ for the reflexive and transitive closure of $\xrightarrow{\tau}$. Moreover, we write $P \xrightarrow{\tau} \equiv Q$ when there exists a process R such that $P \xrightarrow{\tau} R$ and $R \equiv Q$.

The following propositions connect the commitment relation with some of the formal notions of Section 4: exhibiting a barb, reaction, and testing.

PROPOSITION 2. *$P \downarrow \beta$ if and only if there exists an agent A such that $P \xrightarrow{\beta} A$.*

PROPOSITION 3. *$P \to Q$ if and only if $P \xrightarrow{\tau} \equiv Q$.*

PROPOSITION 4. P passes a test (R, β) if and only if there exist an agent A and a process Q such that $P \mid R \xrightarrow{\tau}{}^* Q$ and $Q \xrightarrow{\beta} A$.

The proofs of these propositions are in Appendix A.

5.2. Some Auxiliary Equivalences

In this section, we describe several equivalences on processes that approximate testing equivalence. In particular, in Section 5.2.3, we define barbed congruence, which is a stronger relation than testing equivalence but is sometimes easier to prove directly.

5.2.1. *Strong bisimilarity.* We first recall the definition of strong bisimulation [Mil95]. If $\mathcal{R}$ is a relation on closed processes, we define the relation $\overline{\mathcal{R}}$ on closed agents:

$$
\begin{array}{rcll}
P & \overline{\mathcal{R}} & Q & \text{iff } P \mathcal{R} Q \\
(x)\,P & \overline{\mathcal{R}} & (x)\,Q & \text{iff } P[M/x] \mathcal{R}\, Q[M/x] \text{ for all closed } M \\
(\nu \vec{n})\langle M \rangle\, P & \overline{\mathcal{R}} & (\nu \vec{m})\langle M \rangle\, Q & \text{iff } \vec{m} \text{ is a permutation of } \vec{n} \text{ and } P \mathcal{R} Q
\end{array}
$$

A *strong simulation* is a binary relation $\mathcal{S} \subseteq Proc \times Proc$ such that if $P \mathcal{S} Q$ and $P \xrightarrow{\alpha} A$ then there exists B with $Q \xrightarrow{\alpha} B$ and $A \overline{\mathcal{S}} B$. A relation $\mathcal{S}$ is a *strong bisimulation* if and only if both $\mathcal{S}$ and its converse $\mathcal{S}^{-1}$ are strong simulations.

Strong bisimilarity, written $\sim_s$, is the greatest strong bisimulation, namely the union of all strong bisimulations. Strong bisimilarity is a rather fine-grained equivalence for the spi calculus. For instance, it discriminates between the processes $(\nu K)\, \bar{c}\langle \{M\}_K \rangle$ and $(\nu K)\, \bar{c}\langle \{M'\}_K \rangle$, which we would wish to equate as we explained in Section 3.1. Still, strong bisimilarity is often useful in justifying particular steps of our proofs.

5.2.2. *Barbed equivalence.* Intuitively, one way of weakening strong bisimilarity is to ignore what messages are sent on what channels, and to record only what channels are used. This informal idea leads to the concepts defined here and in Section 5.2.3.

A *barbed simulation* is a binary relation $\mathcal{S} \subseteq Proc \times Proc$ such that $P \mathcal{S} Q$ implies:

(1) for each barb β, if $P \downarrow \beta$ then $Q \downarrow \beta$, and

(2) if $P \to P'$ then there exists Q' such that $Q \to Q'$ and $P' \equiv \mathcal{S} \equiv Q'$

where $P' \equiv \mathcal{S} \equiv Q'$ means that there exist P'' and Q'' such that $P' \equiv P''$, $P'' \mathcal{S} Q''$, and $Q'' \equiv Q'$. A *barbed bisimulation* is a relation $\mathcal{S}$ such that both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are barbed simulations.

Barbed equivalence, written $\dot{\sim}$, is the greatest barbed bisimulation. We prove the following basic facts about barbed equivalence in Appendix C:

PROPOSITION 5. (1) *Barbed equivalence is reflexive, transitive, and symmetric.*

(2) *Structural equivalence implies barbed equivalence.*

(3) *Strong bisimilarity implies barbed equivalence.*

(4) *Barbed equivalence is preserved by restriction.*

It follows from these facts, in particular, that if $P \dot{\sim} Q$ and $P \to P'$ then there exists Q' such that $Q \to Q'$ and $P' \dot{\sim} Q'$.

In order to establish a barbed equivalence, it is often convenient to use Milner's standard technique of "bisimulation up to" [Mil89, MPW92]. A *barbed simulation up to* $\dot{\sim}$ is a binary relation $\mathcal{S} \subseteq Proc \times Proc$ such that $P \mathcal{S} Q$ implies

(1) for each barb β, if $P \downarrow \beta$ then $Q \downarrow \beta$, and

(2) if $P \to P'$ then there exists Q' such that $Q \to Q'$ and $P' \dot{\sim} \mathcal{S} \dot{\sim} Q'$,

where $P' \dot{\sim} \mathcal{S} \dot{\sim} Q'$ means that there exist P'' and Q'' such that $P' \dot{\sim} P''$, $P'' \mathcal{S} Q''$, and $Q'' \dot{\sim} Q'$. A *barbed bisimulation up to* $\dot{\sim}$ is a relation $\mathcal{S}$ such that both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are barbed simulations up to $\dot{\sim}$.

More generally, a *barbed simulation up to* $\dot{\sim}$ *and restriction* is a binary relation $\mathcal{S} \subseteq Proc \times Proc$ such that $P \mathcal{S} Q$ implies:

(1) for each barb β, if $P \downarrow \beta$ then $Q \downarrow \beta$, and

(2) if $P \to P'$ then there exists Q' such that $Q \to Q'$, and there exist P'', Q'', and names $\vec{n}$ such that $P' \dot{\sim} (\nu \vec{n})\, P''$, $Q' \dot{\sim} (\nu \vec{n})\, Q''$, and $P'' \mathcal{S} Q''$.

A *barbed bisimulation up to* $\dot{\sim}$ *and restriction* is a relation $\mathcal{S}$ such that both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are barbed simulations up to $\dot{\sim}$ and restriction.

PROPOSITION 6. *If $\mathcal{S}$ is a barbed bisimulation up to $\dot{\sim}$ and restriction, then $\mathcal{S} \subseteq \dot{\sim}$. A fortiori, if $\mathcal{S}$ is a barbed bisimulation up to $\dot{\sim}$, then $\mathcal{S} \subseteq \dot{\sim}$.*

The proof of this proposition is in Appendix C.

Barbed equivalence is still only a stepping stone. One reason for this is that there are processes that are barbed equivalent but not strongly bisimilar or testing equivalent, such as $\bar{m}\langle n\rangle.\bar{m}\langle n\rangle.\mathbf{0}$ and $\bar{m}\langle n\rangle.\mathbf{0}$, which have the barb $\bar{m}$ and no reactions. Moreover, barbed equivalence is far from being a congruence: it is not even closed under composition, as can be seen by comparing $(\bar{m}\langle n\rangle.\bar{m}\langle n\rangle.\mathbf{0}) \mid (m(x).\mathbf{0})$ and $(\bar{m}\langle n\rangle.\mathbf{0}) \mid (m(x).\mathbf{0})$.

5.2.3. *Barbed congruence.* Barbed congruence, written $\sim$, is the relation on *Proc* obtained by strengthening barbed equivalence as follows:

$$P \sim Q \triangleq \forall R \in Proc(P \mid R \dot{\sim} P \mid R)$$

Unlike barbed equivalence, barbed congruence implies testing equivalence. Therefore, whenever one wishes to prove a testing equivalence (for instance, a secrecy equation), it suffices to prove a barbed congruence. We establish the following properties of barbed congruence in Appendix C:

PROPOSITION 7. (1) *Barbed congruence is reflexive, transitive, and symmetric.*

(2) *Barbed congruence is a congruence on closed processes.*

(3) *Structural equivalence implies barbed congruence.*

(4) *Strong bisimilarity implies barbed congruence.*

(5) *Barbed congruence implies testing equivalence.*

The converses of the implications in parts (3), (4), and (5) do not hold, as we show next.

That barbed congruence does not imply structural equivalence should be fairly evident. We prove it by first establishing a general property of barbed congruence. Let us say that a closed process P is stuck if and only if there is no α and A such that $P \xrightarrow{\alpha} A$. In other words, P is *stuck* if and only if it has no reactions and no barbs.

PROPOSITION 8. *If P is stuck then $P \sim \mathbf{0}$.*

Proof. Assuming that P is stuck, we need to show that $P \mid R \dot{\sim} \mathbf{0} \mid R$ for any closed process R. This holds because any barb or reaction of $P \mid R$ must be due to R alone. ∎

This proposition implies, for example,

$$case\ M\ of\ \{x\}_K\ in\ P \sim \begin{cases} P[N/x] & \text{if } M = \{N\}_K \text{ for some } N \\ \mathbf{0} & \text{otherwise} \end{cases}$$

since *case M of $\{x\}_K$ in P* is stuck unless M is a ciphertext encrypted with K. Since none of the rules of structural equivalence allows us to derive *case M of $\{x\}_K$ in $P \equiv \mathbf{0}$*, barbed congruence does not imply structural equivalence.

Second, barbed congruence does not imply strong bisimilarity. For instance, the processes $(\nu K)\,\bar{c}\langle\{M\}_K\rangle$ and $(\nu K)\,\bar{c}\langle\{M'\}_K\rangle$ are not strongly bisimilar, but they are barbed congruent (as we prove in Section 5.3).

Third, testing equivalence does not imply barbed congruence. Setting

$$\tau.P \triangleq (vm)(\bar{m}\langle * \rangle \mid m(x).P)$$

for $m \notin fn(P)$, $x \notin fv(P)$, we obtain the testing equivalence $P \simeq \tau.P$. (We prove this equivalence in Appendix C.) On the other hand, $P \sim \tau.P$ does not hold in general. Moreover, barbed congruence is more sensitive to the branching structure of processes than testing equivalence.

5.3. *The Underpinning Relation*

In order to reason about attackers and their knowledge, we introduce the underpinning relation. We say that $x_1: \{-\}_{p_1}, ..., x_n: \{-\}_{p_n}$ *underpins* the agent A roughly if A is an agent that may contain occurrences of any of the variables x_1, ..., x_n, but no occurrences of any of the names p_1, ..., p_n. We write this:

$$x_1: \{-\}_{p_1}, ..., x_n: \{-\}_{p_n} \vdash A$$

Our intention is that the variables x_1, ..., x_n represent ciphertexts that an attacker may have intercepted encrypted under the keys p_1, ..., p_n, which the attacker does not have; A, or a subprocess of A, represents the attacker. (Here we take all keys to be names as this suffices for our present purposes; but the general case, where a key is an arbitrary term, could also be interesting.)

Next we give a formal definition of the underpinning relation. A *cipher environment* E is a finite list of entries of the form $x: \{-\}_n$, where x is a variable and n is a name; all the variables must be distinct (but the names need not be). We let $dom(E)$ be the set of variables mentioned in the entries in E, and $keys(E)$ be the set of names mentioned in the entries in E. When E is a cipher environment, M a term, and A an agent, we define:

$$E \vdash M \quad \text{iff} \quad fv(M) \subseteq dom(E) \text{ and } fn(M) \cap keys(E) = \varnothing$$

$$E \vdash A \quad \text{iff} \quad fv(A) \subseteq dom(E) \text{ and } fn(A) \cap keys(E) = \varnothing$$

The relation $\vdash$ is the underpinning relation.

When $x: \{-\}_n$ occurs in a cipher environment, we intend that x stands for a ciphertext of the form $\{M\}_n$. An *E-closure* is a substitution that fixes all the variables in E to appropriate ciphertexts; more precisely, an E-closure is a substitution σ of closed ciphertexts for variables such that $E \vdash \sigma$ is derivable from the following rules:

(Closure $\varnothing$)

$$\frac{}{\varnothing \vdash \varnothing}$$

(Closure Under)

$$\frac{E \vdash \sigma \quad x \notin dom(E) \quad fv(M) = \varnothing}{E, x: \{-\}_n \vdash \sigma, \{M\}_n/x}$$

where $\varnothing$ represents the empty environment, the empty substitution, and the empty set, and $\sigma, \{M\}_n/x$ is the extension of σ that maps x to $\{M\}_n$.

To prove secrecy properties, we would like to show that a process underpinned by a cipher environment acts uniformly no matter which ciphertexts are substituted for the variables in the environment. At first sight one might think that if $E \vdash P$, $E \vdash \sigma$, and $E \vdash \sigma'$, then $P\sigma \sim P\sigma'$ on the reasoning that, since P cannot unwrap the ciphertexts in σ or σ', it will behave the same whether closed by one or the other E-closure. This would hold were it not for the presence of matching in the language. For example, $E = x: \{-\}_m,\ y: \{-\}_m,\ P = [x \text{ is } y]\,\bar{p}\langle 0\rangle,\ \sigma = [\{0\}_m/x, \{0\}_m/y]$, and $\sigma' = [\{0\}_m/x, \{\underline{1}\}_m/y]$ meet the conditions above, but $P\sigma$ may output 0 whereas $P\sigma'$ is stuck. Thus, P can act contingently on the ciphertexts even though it cannot decrypt them. However, if we insist that σ and σ' be injective (that is, $x = y$ whenever $x\sigma = y\sigma$, and similarly for σ') then we obtain $P\sigma \sim P\sigma'$.

These informal arguments lead to the following results.

LEMMA 9. *Suppose that $E \vdash P$ and $E \vdash \sigma$, and that σ is injective.*

(1) *If $P\sigma > Q'$ then there is a process Q with $E \vdash Q$, $fv(Q) \subseteq fv(P)$, $fn(Q) \subseteq fn(P)$, and $Q' = Q\sigma$ such that, whenever $E \vdash \sigma'$ and σ' is injective, $P\sigma' > Q\sigma'$.*

(2) *If $P\sigma \xrightarrow{\alpha} A'$ then there is an agent A with $E \vdash A$, $fv(A) \subseteq fv(P)$, $fn(A) \subseteq fn(P)$, and $A' = A\sigma$ such that, whenever $E \vdash \sigma'$ and σ' is injective, $P\sigma' \xrightarrow{\alpha} A\sigma'$.*

The proof of this lemma is in Appendix D.

PROPOSITION 10. *Suppose that $E \vdash \sigma$ and $E \vdash \sigma'$, and that both σ and σ' are injective. Then $\mathscr{S} = \{(P\sigma, P\sigma') \mid E \vdash P\}$ is a barbed bisimulation.*

Proof. Consider any commitment $P\sigma \xrightarrow{\alpha} A'$. By Lemma 9, there is an agent A with $E \vdash A$, $A' = A\sigma$, and $P\sigma' \xrightarrow{\alpha} A\sigma'$. Therefore, any barb of $P\sigma$ is also exhibited by $P\sigma'$, and any reaction of $P\sigma$ may be matched up to $\mathscr{S}$ by $P\sigma'$. Therefore, $\mathscr{S}$ is a barbed simulation. Indeed by symmetry it is a barbed bisimulation. ∎

This last proposition provides an easy way to prove some equivalences, as we now demonstrate with a small proof of a familiar secrecy property. We prove that, for any closed terms M and M':

$$(\nu K)\,\bar{c}\langle \{M\}_K \rangle \sim (\nu K)\,\bar{c}\langle \{M'\}_K \rangle$$

By (Struct Extrusion) and Proposition 5, it suffices to prove that:

$$\bar{c}\langle \{M\}_K \rangle \mid R \;\dot{\sim}\; \bar{c}\langle \{M'\}_K \rangle \mid R$$

for any R such that $K \notin fn(R)$. But this follows from Proposition 10 with $E = x: \{-\}_K$, $P = \bar{c}\langle x\rangle \mid R$, $\sigma = [\{M\}_K/x]$, and $\sigma' = [\{M'\}_K/x]$.

6. PROOFS FOR THE EXAMPLES

Having defined the semantics of the spi calculus and developed some proof techniques, we revisit the examples of the first half of the paper. We prove most of the authenticity and secrecy properties claimed in those examples. Additional proofs appear in a technical report [AG97a]. Our proofs are not quite as easy as those of special-purpose formalisms (e.g., [BAN89]), but they have a somewhat clearer status. With a few further techniques and tools, proofs such as ours could well become routine.

6.1. Proofs for the Example of Section 2.3.1

The example of Section 2.3.1 is our simplest one. We can prove the authenticity property $Inst(M) \simeq Inst_{spec}(M)$ by using strong bisimilarity.

PROPOSITION 11. *For any closed term* M, $Inst(M) \simeq Inst_{spec}(M)$.

Proof. The only commitments of $Inst(M)$ and $Inst_{spec}(M)$ are:

$$Inst(M) \xrightarrow{\tau} (\nu c_{AB})(\mathbf{0} \mid F(M))$$

$$Inst_{spec}(M) \xrightarrow{\tau} (\nu c_{AB})(\mathbf{0} \mid F(M))$$

It follows that $Inst(M) \sim_s Inst_{spec}(M)$, that $Inst(M) \sim Inst_{spec}(M)$ (by Proposition 7(4)), and finally that $Inst(M) \simeq Inst_{spec}(M)$ (by Proposition 7(5)). ∎

Turning to secrecy, we first prove a restricted version of the secrecy property claimed in Section 2.3.1:

LEMMA 12. *$Inst(M) \simeq Inst(M')$ if $F(x)$ is $\bar{c}\langle * \rangle$, for any closed terms M and M'.*

Proof. For any N, the only commitment of $Inst(N)$ is:

$$Inst(N) \xrightarrow{\tau} (\nu c_{AB})(\mathbf{0} \mid \bar{c}\langle * \rangle)$$

so clearly $Inst(M) \sim_s Inst(M')$. As in the previous proof, $Inst(M) \simeq Inst(M')$ follows. ∎

Now a little calculation yields the full secrecy property:

PROPOSITION 13. *$Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for any closed terms M and M'.*

Proof. Let us write $Inst(M, (x)\, \bar{c}\langle * \rangle)$ for $Inst(M)$ in the special case where $F(x)$ is $\bar{c}\langle * \rangle$ (as in Lemma 12); note that in this case $Inst(M)$ and $Inst_{spec}(M)$ are literally identical.

Assuming that c is a fresh name and y a fresh variable, we write $\tau.F(N)$ for $(\nu c)(\bar{c}\langle * \rangle \mid c(y).F(N))$. For any closed N, we have

$$(\nu c)(c_{AB}(x).\bar{c}\langle * \rangle \mid c(y).F(N)) \sim_s c_{AB}(x).\tau.F(N)$$

because the only commitments of these processes are:

$$(vc)(c_{AB}(x).\bar{c}\langle * \rangle \mid c(y).F(N)) \xrightarrow{c_{AB}} (x)\,\tau.F(N)$$
$$c_{AB}(x).\tau.F(N) \xrightarrow{c_{AB}} (x)\,\tau.F(N)$$

Hence we obtain the equation:

$$Inst_{spec}(N) \simeq (vc)(Inst(N, (x)\,\bar{c}\langle * \rangle) \mid c(y).F(N)) \tag{1}$$

as follows:

$$\begin{aligned}
Inst_{spec}(N) &= (vc_{AB})(\overline{c_{AB}}\langle N \rangle.\mathbf{0} \mid c_{AB}(x).F(N)) \\
&\simeq (vc_{AB})(\overline{c_{AB}}\langle N \rangle.\mathbf{0} \mid c_{AB}(x).(\tau.F(N))) \\
&\simeq (vc_{AB})(\overline{c_{AB}}\langle N \rangle.\mathbf{0} \mid (vc)(c_{AB}(x).\bar{c}\langle * \rangle \mid c(y).F(N))) \\
&\equiv (vc)(vc_{AB})(\overline{c_{AB}}\langle N \rangle.\mathbf{0} \mid c_{AB}(x).\bar{c}\langle * \rangle) \mid c(y).F(N)) \\
&= (vc)(Inst(N, (x)\,\bar{c}\langle * \rangle) \mid c(y).F(N))
\end{aligned}$$

making use of the "τ law" $F(N) \simeq \tau.F(N)$ (Proposition 32), and of the facts that testing equivalence is a congruence (Proposition 1) and that strong bisimilarity implies testing equivalence (Proposition 7).

Finally, Eq. (1), Lemma 12, the authenticity property of Proposition 11, and the assumption $F(M) \simeq F(M')$ justify the following calculation:

$$\begin{aligned}
Inst(M) &\simeq Inst_{spec}(M) \\
&\simeq (vc)(Inst(M, (x)\,\bar{c}\langle * \rangle) \mid c(y).F(M)) \\
&\simeq (vc)(Inst(M', (x)\,\bar{c}\langle * \rangle) \mid c(y).F(M')) \\
&\simeq Inst_{spec}(M') \\
&\simeq Inst(M') \quad \blacksquare
\end{aligned}$$

6.2. Proofs for the Example of Section 3.2.1.

For the example of Section 2.3.1, which does not use cryptography, the proof of authenticity is simply a proof of strong bisimilarity. We cannot proceed analogously for the example of Section 3.2.1, because in fact $Inst(M)$ and $Inst_{spec}(M)$ are not strongly bisimilar; instead, we prove that $Inst(M)$ and $Inst_{spec}(M)$ are barbed congruent.

PROPOSITION 14. *For any closed term* M, $Inst(M) \simeq Inst_{spec}(M)$.

Proof. We prove that $Inst(M) \sim Inst_{spec}(M)$; the claim then follows since barbed congruence implies testing equivalence according to Proposition 7.

Suppose that R is some arbitrary closed process and M is some arbitrary closed term. Without loss of generality, we assume that $K_{AB} \notin fn(R)$. Below we show that:

$$(\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B \mid R) \dot{\sim} (\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B_{spec}(M) \mid R) \tag{2}$$

By Proposition 5(4), it follows that:

$$(\nu K_{AB})(\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B \mid R) \dot{\sim} (\nu K_{AB})(\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B_{spec}(M) \mid R)$$

Since $K_{AB} \notin fn(R)$, we have:

$$Inst(M) \mid R \equiv (\nu K_{AB})(\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B \mid R)$$

and similarly:

$$Inst_{spec}(M) \mid R \equiv (\nu K_{AB})(\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B_{spec}(M) \mid R)$$

Since barbed equivalence respects structural equivalence (by Proposition 5), we obtain:

$$Inst(M) \mid R \dot{\sim} Inst_{spec}(M) \mid R$$

By the definition of barbed congruence, we conclude:

$$Inst(M) \sim Inst_{spec}(M)$$

It remains to give a proof of Eq. (2). For this proof, we let $\sigma = [\{M\}_{K_{AB}}/x]$ and introduce the following relation $\mathscr{S}$:

$$P \mathscr{S} Q \quad \text{iff} \quad P = B \mid R_1\sigma \text{ and } Q = B_{spec}(M) \mid R_1\sigma$$
$$\text{for some } R_1 \text{ such that } x\colon \{-\}_{K_{AB}} \vdash R_1$$

Intuitively, the process $R_1\sigma$ represents both A and an attacker that does not have K_{AB}. We prove that $\mathscr{S} \cup \dot{\sim}$ is a barbed bisimulation. This amounts to showing that if $P \mathscr{S} Q$ then P and Q can each match the other's barbs and reactions.

If $P \mathscr{S} Q$ then there exists R_1 such that $P = B \mid R_1\sigma$ and $Q = B_{spec}(M) \mid R_1\sigma$, and $x\colon \{-\}_{K_{AB}} \vdash R_1$. Hence the barbs of P are:

(1) $P \downarrow c_{AB}$ (from B),

(2) $P \downarrow \beta$ if $R_1\sigma \downarrow \beta$.

Clearly Q exhibits these barbs too. The reactions of P are:

(1) if $R_1\sigma \xrightarrow{\overline{c_{AB}}} (\nu \vec{n})\langle N \rangle\, R'$ and $P' \equiv (\nu \vec{n})(case\ N\ of\ \{y\}_{K_{AB}}\ in\ F(y) \mid R')$ then $P \to P'$,

(2) if $R_1\sigma \xrightarrow{\tau} R'$ and $P' \equiv B \mid R'$ then $P \to P'$.

(One can calculate these reactions via the commitment relation and Proposition 3. Without loss of generality, we assume that the names $\vec{n}$ are fresh.) We show that, in each case, Q can match these reactions of P.

(1) One of the reactions of Q is:

$$Q \to Q' \triangleq (\nu\vec{n})(\mathit{case}\ N\ \mathit{of}\ \{y\}_{K_{AB}}\ \mathit{in}\ F(M) \mid R')$$

Now it suffices to show that $P' \mathrel{\dot{\sim}} Q'$. By Lemma 9(2), there exists R'_1 such that $x\colon \{-\}_{K_{AB}} \vdash R'_1$ and $R'_1\sigma = (\nu\vec{n})\langle N\rangle\ R'$. Therefore, R'_1 must have the form $(\nu\vec{n})\langle N_0\rangle\ R_0$ with $N = N_0\sigma$, $R' = R_0\sigma$, and both $x\colon \{-\}_{K_{AB}} \vdash N_0$ and $x\colon \{-\}_{K_{AB}} \vdash R_0$. Since $x\colon \{-\}_{K_{AB}} \vdash N_0$, either $N_0\sigma$ is $\{M\}_{K_{AB}}$ (if N_0 is x) or $N_0\sigma$ is not a ciphertext encrypted with K_{AB}.

In the former case, we have:

$$\begin{aligned} P' &\equiv (\nu\vec{n})(\mathit{case}\ \{M\}_{K_{AB}}\ \mathit{of}\ \{y\}_{K_{AB}}\ \mathit{in}\ F(y) \mid R') \\ &\equiv (\nu\vec{n})(F(M) \mid R') \\ &\equiv (\nu\vec{n})(\mathit{case}\ \{M\}_{K_{AB}}\ \mathit{of}\ \{y\}_{K_{AB}}\ \mathit{in}\ F(M) \mid R') \\ &\equiv Q' \end{aligned}$$

In the latter case, decryption gets stuck, and by appeal to Propositions 5 and 8 we get:

$$\begin{aligned} P' &\equiv (\nu\vec{n})(\mathit{case}\ N\ \mathit{of}\ \{y\}_{K_{AB}}\ \mathit{in}\ F(y) \mid R') \\ &\mathrel{\dot{\sim}} (\nu\vec{n})(\mathbf{0} \mid R') \\ &\mathrel{\dot{\sim}} (\nu\vec{n})(\mathit{case}\ N\ \mathit{of}\ \{y\}_{K_{AB}}\ \mathit{in}\ F(M) \mid R') \\ &\equiv Q' \end{aligned}$$

In both cases, we obtain $P' \mathrel{\dot{\sim}} Q'$ by Proposition 5.

(2) One of the reactions of Q is:

$$Q \to Q' \triangleq B_{spec}(M) \mid R'$$

Now it suffices to show that $P' \mathrel{\dot{\sim}} \mathscr{S} \mathrel{\dot{\sim}} Q'$. By Lemma 9(2), there exists R'_1 such that $x\colon \{-\}_{K_{AB}} \vdash R'_1$ and $R'_1\sigma = R'$. Therefore, $(B \mid R')\ \mathscr{S} Q'$, and hence $P' \equiv \mathscr{S} \equiv Q'$.

Almost identical reasoning shows that P can match the barbs and reactions of Q. We conclude that $\mathscr{S} \cup \mathrel{\dot{\sim}}$ is a barbed bisimulation, so $\mathscr{S} \subseteq \mathrel{\dot{\sim}}$.

In order to derive Eq. (2), we let $R_1 = \overline{c_{AB}}x \mid R$. We obtain:

$$\begin{aligned}\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B \mid R &\equiv B \mid R_1\sigma \\ &\mathscr{S}\ B_{spec}(M) \mid R_1\sigma \\ &\equiv \overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B_{spec}(M) \mid R.\end{aligned}$$

Equation (2) follows since $\mathscr{S} \subseteq \dot{\sim}$ and by Proposition 5. ▮

For proving secrecy, we adopt the same general strategy as in Section 6.1. We first prove a restricted version of the secrecy property:

LEMMA 15. *$Inst(M) \simeq Inst(M')$ if $F(x)$ is $\bar{c}\langle * \rangle$, for any closed terms M and M'.*

Proof. Almost exactly as in the proof of Proposition 14, it suffices to prove the equation

$$(\overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle \mid B \mid R) \ \dot{\sim}\ (\overline{c_{AB}}\langle \{M'\}_{K_{AB}} \rangle \mid B \mid R) \tag{3}$$

for any closed process R such that $K_{AB} \notin fn(R)$, and any closed terms M and M'.

For the proof of this equation, we let $\sigma = [\{M\}_{K_{AB}}/x]$, $\sigma' = [\{M'\}_{K_{AB}}/x]$, and introduce the following relation $\mathscr{S}$

$$\begin{aligned}P \mathscr{S} Q \quad \text{iff} \quad & P = B \mid R_1\sigma \text{ and } Q = B \mid R_1\sigma' \\ & \text{for some } R_1 \text{ such that } x\colon \{-\}_{K_{AB}} \vdash R_1\end{aligned}$$

Much as in the proof of Proposition 14, we can establish that $\mathscr{S} \cup \dot{\sim}$ is a barbed bisimulation. The proof relies on Proposition 10, which implies for example that the relation $\{(R_1\sigma, R_1\sigma') \mid x\colon \{-\}_{K_{AB}} \vdash R_1\}$ is a barbed bisimulation. (See [AG97a] for details.) ▮

The full secrecy property follows.

PROPOSITION 16. *$Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for any closed terms M and M'.*

Proof. The proof is exactly analogous to that of Proposition 13, and relies on Proposition 14, Lemma 15, and the equation

$$Inst_{spec}(N) \simeq (\nu c)(Inst(N, (x)\,\bar{c}\langle * \rangle) \mid c(y).F(N)) \quad ▮$$

6.3. Formalization of the Attack of Section 3.2.3

Here we prove that the authenticity equation discussed in Section 3.2.3 does not hold. We do this by formalizing the replay attack sketched there.

PROPOSITION 17. *If I is (i, j, M), I' is (i, j, M'), and M and M' are different closed terms, then there exists F such that $Sys(I, I') \not\simeq Sys_{spec}(I, I')$.*

Proof. We define $F(x, y, z)$ as $\bar{c}\langle z\rangle$, where c is a new name. According to the definition of testing equivalence, it suffices to construct a test (R, β) such that $Sys(I, I')$ passes (R, β) but $Sys_{spec}(I, I')$ does not pass (R, β).

For β, we take $\bar{d}$ where d is a name that does not occur free in $Sys_{spec}(I, I')$. For R, we take:

$$c_S(u).\overline{c_S}\langle u\rangle.\overline{c_S}\langle u\rangle.c_j(x).\overline{c_j}\langle x\rangle.\overline{c_j}\langle x\rangle.c(y).c(z).[y \text{ is } z]\,\bar{d}\langle *\rangle$$

This process duplicates a message u sent on c_S and a message x sent on c_j, receives two messages y and z through c, and finally sends a message on d if y and z are equal. Intuitively, this process can be understood as an attacker that replays an encrypted key u and some encrypted data x from $\underline{i}$, and signals on d if the replay attack has worked, that is, if two identical messages y and z appear on c.

The parallel composition of R with $Sys(I, I')$ may eventually exhibit $\bar{d}$, because y and z may both equal M or M', as a result of the message duplications on c_S and c_j. Therefore, $Sys(I, I')$ passes (R, β). ∎

In contrast, the parallel composition of R with $Sys_{spec}(I, I')$ never exhibits d, because each of M and M' will be transmitted at most once on c, so y and z cannot match. Therefore, $Sys_{spec}(I, I')$ does not pass (R, β).

6.4. *Proofs for the Example of Section* 3.2.4

As in Section 3.2.4, we consider a system with a server S and n other principals, which we call $\underline{1}$, $\underline{2}$, ... We let $Prn = 1..n$, and we use the metavariables i and j to range over Prn. Each principal has an input channel; these input channels have the names c_1, c_2, ..., c_n and c_S. The server shares a pair of keys with each other principal: principal i uses key K_{iS} to send to S and key K_{Si} to receive from S, for each $i \in Prn$. The system is parameterized by a list of instances, I_1, ..., I_m, indexed by the set $Ins = 1..m$, and a single abstraction F such that $F(\underline{i}, \underline{j}, M)$ is a process for any instance (i, j, M). We use the metavariable k to range over Ins.

We rephrase the formal description of the protocol through the following definitions:

$$A1(i, j, M) \triangleq \overline{c_S}\langle \underline{i}\rangle \mid A2(i, j, M)$$

$$A2(i, j, M) \triangleq c_i(x).(\nu K_{AB})(\overline{c_S}\langle(\underline{i}, \{\underline{i}, \underline{i}, \underline{j}, K_{AB}, x\}_{K_{iS}})\rangle \mid \overline{c_j}\langle(\underline{i}, \{M\}_{K_{AB}})\rangle)$$

$$S1 \triangleq c_S(x).\prod_{i \in Prn}[x \text{ is } \underline{i}](\nu N_S)(\overline{c_i}\langle N_S\rangle \mid S2(i, N_S))$$

$$S2(i, N) \triangleq c_S(x).let(y_1, y_2) = x \text{ in}$$
$$[y_1 \text{ is } \underline{i}] \text{ case } y_2 \text{ of } \{z_1, z_2, z_3, z_4, z_5\}_{K_{iS}} \text{ in}$$
$$\prod_{j \in Prn}[z_1 \text{ is } \underline{i}][z_2 \text{ is } \underline{i}][z_3 \text{ is } \underline{j}][z_5 \text{ is } N]\, S3(i, j, z_4)$$

$$S3(i, j, K) \triangleq \overline{c_j}\langle *\rangle \mid S4(i, j, K)$$

$$S4(i, j, K) \triangleq c_S(x).\overline{c_j}\langle\{S, \underline{i}, \underline{j}, K, x\}_{K_{Sj}}\rangle$$

$$B1(j, F) \triangleq c_j(x).(\nu N_B)(\overline{c_S}\langle N_B\rangle \mid B2(j, F, N_B))$$

$$B2(j, F, N) \triangleq c_j(x).case\ x\ of\ \{y_1, y_2, y_3, y_4, y_5\}_{K_{Sj}}\ in$$
$$\prod_{i \in Prn} [y_1\ is\ S][y_2\ is\ \underline{i}][y_3\ is\ \underline{j}][y_5\ is\ N]\ B3(i, j, F, y_4)$$

$$B3(i, j, F, K) \triangleq c_j(x).let(y_1, y_2) = x\ in$$
$$[y_1\ is\ \underline{i}]case\ y_2\ of\ \{z\}_K\ in\ F(\underline{i}, \underline{j}, z)$$

$$Sys(I_1, ..., I_m) \triangleq (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn})$$
$$\left(\prod_{k \in Ins} A1(I_k) \mid !S1 \mid \prod_{j \in Prn} !B1(j, F)\right)$$

where $(\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn})$ means $(\nu K_{1S})\ldots(\nu K_{nS})(\nu K_{S1})\ldots(\nu K_{Sn})$.

We rephrase the specification as well:

$$A1_{spec}((i, j, M), F) \triangleq (\nu p)(A1(i, j, p) \mid p(x).F(\underline{i}, \underline{j}, M))$$

$$F_{spec}(i, j, p) \triangleq \bar{p}\langle *\rangle$$

$$Sys_{spec}(I_1, ..., I_m) \triangleq (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn})$$
$$\left(\prod_{k \in Ins} A1_{spec}(I_k, F) \mid !S1 \mid \prod_{j \in Prn} !B1(j, F_{spec})\right)$$

PROPOSITION 18. *For any instances* $I_1, ..., I_m$,

$$Sys(I_1, ..., I_m) \simeq Sys_{spec}(I_1, ..., I_m)$$

Proof. Let $I_1, ..., I_m$ be a list of instances, with $Ins = 1..m$. We begin by reducing the problem to one involving finite compositions rather than replications, and give a bisimulation proof after this reduction.

First, we group the replications in $Sys(I_1, ..., I_m)$ and $Sys_{spec}(I_1, ..., I_m)$ using Proposition 27,

$$Sys(I_1, ..., I_m) \simeq (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn})$$
$$\left(\prod_{k \in Ins} A1(I_k) \;\middle|\; !\left(S1 \;\middle|\; \prod_{j \in Prn} B1(j, F)\right)\right) \tag{4}$$

$$Sys_{spec}(I_1, ..., I_m) \simeq (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn})$$
$$\left(\prod_{k \in Ins} A1_{spec}(I_k, F) \;\middle|\; !\left(S1 \;\middle|\; \prod_{j \in Prn} B1(j, F_{spec})\right)\right) \tag{5}$$

Further, we apply Proposition 26 to the right-hand sides of (4) and (5); Proposition 26 implies that, to prove $Sys(I_1, ..., I_m) \simeq Sys_{spec}(I_1, ..., I_m)$, it suffices to prove

$$fSys(I_1, ..., I_m, r) \simeq fSys_{spec}(I_1, ..., I_m, r)$$

for all $r \geqslant 0$, where

$$\begin{aligned} fSys(I_1, ..., I_m, r) \triangleq\ & (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn}) \\ & \left(\prod_{k \in Ins} A1(I_k) \,\middle|\, \prod_{s \in 1..r} \left(S1 \,\middle|\, \prod_{j \in Prn} B1(j, F) \right) \right) \\ fSys_{spec}(I_1, ..., I_m, r) \triangleq\ & (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn}) \\ & \left(\prod_{k \in Ins} A1_{spec}(I_k, F) \,\middle|\, \prod_{s \in 1..r} \left(S1 \,\middle|\, \prod_{j \in Prn} B1(j, F_{spec}) \right) \right) \end{aligned}$$

Thus, we have eliminated replications.

Next we reformulate (6) by pulling restrictions to the top level, and inserting certain additional τ steps. For this purpose, we use the following auxiliary definitions:

$$\begin{aligned} A1'((i, j, M), K) &\triangleq \overline{c_S}\langle \underline{i} \rangle \mid A2'((i, j, M), K) \\ A2'((i, j, M), K) &\triangleq c_i(x).(\overline{c_S}\langle (\underline{i}, \{\underline{i}, \underline{i}, \underline{j}, K, x\}_{K_{iS}}) \rangle \mid \overline{c_j}\langle (i, \{M\}_K) \rangle) \\ S1'(N) &\triangleq c_S(x). \prod_{i \in Prn} [x \text{ is } \underline{i}](\overline{c_i}\langle N \rangle \mid S2(i, N)) \\ B1'(j, F, N) &\triangleq c_j(x).(\overline{c_S}\langle N \rangle \mid B2(j, F, N)) \end{aligned}$$

Lemmas 33 and 34 yield:

$$A1(I) \simeq (\nu K_{AB})\, A1'(I, K_{AB}) \tag{7}$$

$$A2(I) \simeq (\nu K_{AB})\, A2'(I, K_{AB}) \tag{8}$$

$$S1 \simeq (\nu N_S)\, S1'(N_S) \tag{9}$$

$$B1(j, F) \simeq (\nu N_B)\, B1'(j, F, N_B) \tag{10}$$

Moreover, Eq. (7) yields:

$$A1_{spec}(i, j, M) \simeq (\nu K_{AB})(\nu p)(A1'((i, j, p), K_{AB}) \mid p(x).F(\underline{i}, \underline{j}, M)) \tag{11}$$

We also introduce the sets of names:

$$\{p_k \mid k \in Ins\}$$
$$\{K_{ABk} \mid k \in Ins\}$$
$$\{N_{Ss} \mid s \in 1..r\}$$
$$\{N_{Bjt} \mid j \in Prn \;\&\; t \in 1..r\}$$

All the names listed are assumed distinct and fresh. Given that $\tau.F$ is short for the abstraction $(x)\,\tau.F(x)$, we obtain:

$$\begin{aligned} fSys(I_1, ..., I_m, r) \simeq\; & (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn}) \\ & (\nu K_{ABk}{}^{k \in Ins})(\nu N_{Ss}{}^{s \in 1..r})(\nu_{Bjt}{}^{j \in Prn \,\&\, t \in 1..r}) \\ & \Big(\prod_{k \in Ins} A1'(I_k, K_{ABk}) \;\Big|\; \prod_{s \in 1..r} S1'(N_{Ss}) \;\Big| \\ & \prod_{j \in Prn} \prod_{t \in 1..r} B1'(j, \tau.F, N_{Bjt}) \Big) \end{aligned} \tag{12}$$

$$\begin{aligned} fSys_{spec}(I_1, ..., I_m, r) \simeq\; & (\nu K_{iS}{}^{i \in Prn})(\nu K_{Sj}{}^{j \in Prn})(\nu p_k{}^{k \in Ins}) \\ & (\nu K_{ABk}{}^{k \in Ins})(\nu N_{Ss}{}^{s \in 1..r})(\nu N_{Bjt}{}^{j \in Prn \,\&\, t \in 1..r}) \\ & \Big(\Big(\prod_{k \in Ins} A1'((i, j, p_k), K_{ABk}) \\ & \quad \text{where} \quad I_k = (i, j, M) \Big) \;\Big| \\ & \Big(\prod_{k \in Ins} p_k(x).F(\underline{i}, \underline{j}, M) \\ & \quad \text{where } I_k = (i, j, M) \Big) \;\Big| \\ & \prod_{s \in 1..r} S1'(N_{Ss}) \;\Big| \\ & \prod_{j \in Prn} \prod_{t \in 1..r} B1'(j, F_{spec}, N_{Bjt}) \Big) \end{aligned} \tag{13}$$

The proof of (12) and (13) is in three steps. First, we expose all the restrictions in the processes $fSys(I_1, ..., I_m, r)$ and $fSys_{spec}(I_1, ..., I_m, r)$ by rewriting with Eqs. (7), (8), (9), (10), and (11). Second, we use the rules of structural equivalence to group all the restrictions at the top level of the processes. Third, we use the τ law ($\tau.P \simeq P$, Proposition 32) to insert a τ step before each call to F in $fSys(I_1, ..., I_m, r)$. (The τ step is useful because it corresponds to the interaction on one of the p_k's that precedes each call to F in $fSys_{spec}(I_1, ..., I_m, r)$.)

Thus, we have reduced the property claimed in this proposition to Eq. (6), and in turn have reduced this equation to the equivalence of the right-hand sides of Eqs. (12) and (13), for an arbitrary number $r \geqslant 0$. To prove this equivalence, we invoke Proposition 7, and show that when composed with any closed process R the two right-hand sides of (12) and (13) are barbed equivalent. Without loss of generality we may assume that none of the names bound in the outermost restrictions occurs free in R. Up to structural equivalence, and therefore barbed equivalence, we may extrude the scope of those restrictions to include R. Since barbed equivalence is preserved by restriction (Proposition 5(4)), it suffices to prove that the following two processes are barbed equivalent:

$$\begin{array}{l} \left(\prod_{k \in Ins} A1'(I_k, K_{ABk}) \right| \\ \left. \prod_{s \in 1..r} S1'(N_{Ss}) \right| \prod_{j \in Prn} \prod_{t \in 1..r} B1'(j, \tau.F, N_{Bjt}) \Big| R \end{array} \tag{14}$$

and

$$\begin{array}{l} (\nu p_k{}^{k \in Ins}) \\ \left(\left(\prod_{k \in Ins} A1'((i, j, p_k), K_{ABk}) \text{ where } I_k = (i, j, M) \right) \right| \\ \left(\prod_{k \in Ins} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_k = (i, j, M) \right) \Big| \\ \left. \prod_{s \in 1..r} S1'(N_{Ss}) \right| \prod_{j \in Prn} \prod_{t \in 1..r} B1'(j, F_{spec}, N_{Bjt}) \Big| R \Big) \end{array} \tag{15}$$

for any closed R such that no K_{iS}, K_{Sj}, K_{ABk}, N_{Ss}, N_{Bjt}, or p_k occurs free in R. (We have removed most of the outermost restrictions only for the sake of notational simplicity. On the other hand, it is necessary to retain the restriction on the p_k's: otherwise the simplified process (15) would have input barbs p_k that could not be matched by process (14).)

The remainder of our proof consists in constructing a relation $\mathscr{S}$ such that $\equiv \mathscr{S} \equiv$ relates processes (14) and (15), and in establishing that $\mathscr{S}$ is a barbed bisimulation up to $\dot{\sim}$ and restriction, hence that processes (14) and (15) are barbed equivalent. We lead up to the definition of $\mathscr{S}$ with several preliminary definitions:

- We let a *world* be a tuple $W = (snd, srv, rcv, X, E, \sigma, \sigma_{spec}, R)$ where E is an environment and σ and σ_{spec} are substitutions, R is a process, $X \subseteq Ins$, and *snd*, *srv*, and *rcv* are finite maps such that:

$$\begin{array}{r} snd(k) \in \{a2, sent(L, L') \mid \text{any closed terms } L \text{ and } L'\} \\ srv(s) \in \{s1, s2(i), stuck, s4(k), sent(k, L, L') \mid \\ i \in Prn, k \in Ins, \text{any closed terms } L \text{ and } L'\} \\ rcv(j, t) \in \{b1, b2, stuck, b3(k), run(k), done \mid k \in Ins\} \end{array}$$

for each $k \in Ins$, $s \in 1..r$, and $(j, t) \in Prn \times 1..r$. The symbols $a2$, *sent*, $s1$, $s2$, *stuck*, $s4$, $b1$, $b2$, *stuck*, $b3$, *run*, and *done* are string tags; $s2(i)$ is short for the pair $(s2, i)$, *sent*(k, L, L') for the pair $(sent, (k, L, L'))$, and similarly for the other tags.

Intuitively, $k \in X$ just if instance k may yet complete the protocol. The maps *snd*, *srv*, and *rcv* represent the states of the senders, servers, and receivers, respectively, that participate in the protocol.

- Given a world $W = (snd, srv, rcv, X, E, \sigma, \sigma_{spec}, R)$, and given $k \in Ins$, $s \in 1..r$, and $(j, t) \in Prn \times 1..r$, we define processes $A^W(k)$, $A^W_{spec}(k)$, $S^W(s)$, $B^W(j, t)$, and $B^W_{spec}(j, t)$:

$$A^W(k) \triangleq \begin{cases} A2'(I_k, K_{ABk}) & \text{if } snd(k) = a2 \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$A^W_{spec}(k) \triangleq \begin{cases} A2'((i, j, p_k), K_{ABk}) & \text{if } snd(k) = a2,\ I_k = (i, j, M) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$S^W(s) \triangleq \begin{cases} S1'(N_{Ss}) & \text{if } srv(s) = s1 \\ S2(i, N_{Ss}) & \text{if } srv(s) = s2(i) \\ S4(i, j, K_{ABk}) & \text{if } srv(s) = s4(k),\ I_k = (i, j, M) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$B^W(j, t) \triangleq \begin{cases} B1'(j, \tau.F, N_{Bjt}) & \text{if } rcv(j, t) = b1 \\ B2(j, \tau.F, N_{Bjt}) & \text{if } rcv(j, t) = b2 \\ B3(i, j, \tau.F, K_{ABk}) & \text{if } rcv(j, t) = b3(k),\ I_k = (i, j, M) \\ \tau.F(\underline{i}, \underline{j}, M) & \text{if } rcv(j, t) = run(k),\ I_k = (i, j, M) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$B^W_{spec}(j, t) \triangleq \begin{cases} B1'(j, F_{spec}, N_{Bjt}) & \text{if } rcv(j, t) = b1 \\ B2(j, F_{spec}, N_{Bjt}) & \text{if } rcv(j, t) = b2 \\ B3(i, j, F_{spec}, K_{ABk}) & \text{if } rcv(j, t) = b3(k),\ I_k = (i, j, M) \\ \overline{p_k}\langle * \rangle & \text{if } rcv(j, t) = run(k),\ I_k = (i, j, M) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Intuitively, $A^W(k)$ is the process that sender k has left to run when its state is *snd*(k). Similarly, in the context of the specification, $A^W_{spec}(k)$ is the process that sender k has left to run when its state is *snd*(k); this process does not include $p_k(x).F(\underline{i}, \underline{j}, M)$, which is treated separately. The other definitions deal analogously with replicas of the server and of the receivers.

Given a world $W = (snd, srv, rcv, X, E, \sigma, \sigma_{spec}, R)$, we also let P^W be:

$$\prod_{k \in Ins} A^W(k) \;\Big|\; \prod_{s \in 1..r} S^W(s) \;\Big|\; \prod_{(j, t) \in Prn \times 1..r} B^W(j, t) \;\Big|\; R\sigma$$

and Q^W be:

$$(\nu p_k{}^{k \in Ins}) \left(\prod_{k \in Ins} A^W_{spec}(k) \,\middle|\, \prod_{s \in 1..r} S^W(s) \,\middle|\, \prod_{(j,t) \in Prn \times 1..r} B^W_{spec}(j,t) \,\middle|\, \left(\prod_{k \in X} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_k = (i,j,M) \right) \,\middle|\, R\sigma_{spec} \right)$$

Intuitively, P^W is the process that the whole system has left to run when its state is as described in W, and Q^W is the corresponding process for the specification.

- Given a world W with maps *snd*, *srv*, and *rcv*, we define the *instance sets* of W to be the subsets X_2^W, X_3^W, X_5^W, X_6^W, X_7^W, X_8^W of *Ins*, such that for any $k \in Ins$ with $I_k = (i, j, M)$:

$$k \in X_2^W \quad \text{iff} \quad snd(k) = a2$$

$$k \in X_3^W \quad \text{iff} \quad \exists s \in 1..r, i' \in Prn(snd(k) = sent(N_{Ss}, N_{Ss}) \;\&\; srv(s) \in \{s1, s2(i')\})$$

$$k \in X_5^W \quad \text{iff} \quad \exists s \in 1..r(srv(s) = s4(k))$$

$$k \in X_6^W \quad \text{iff} \quad \exists s \in 1..r, t \in 1..r \; (srv(s) = sent(k, N_{Bjt}, N_{Bjt}) \;\&\; rcv(j,t) \in \{b1, b2\})$$

$$k \in X_7^W \quad \text{iff} \quad \exists t \in 1..r(rcv(j,t) = b3(k))$$

$$k \in X_8^W \quad \text{iff} \quad \exists t \in 1..r(rcv(j,t) = run(k))$$

Intuitively, if $k \in X_s^W$ and $s \in \{2, 3, 5, 6, 7\}$, then the message in the protocol numbered s is the next to be received in instance k. Instance set X_8^W represents instances that, having completed the protocol, are a τ step away from running F.

- A world $W = (snd, srv, rcv, X, E, \sigma, \sigma_{spec}, R)$ is *possible* if and only if the following conditions hold:

(1) Sets X_2^W, X_3^W, X_5^W, X_6^W, X_7^W, X_8^W are pairwise disjoint.

(2) The union $X_2^W \cup X_3^W \cup X_5^W \cup X_6^W \cup X_7^W \cup X_8^W$ is a subset of X.

(3) For any $k \in Ins$, $s \in 1..r$, and terms L and L', if either $srv(s) = s4(k)$ or $srv(s) = sent(k, L, L')$ then $snd(k) = sent(N_{Ss}, N_{Ss})$.

(4) For any $k \in Ins$, $j \in Prn$, and $t \in 1..r$, if either $rcv(j,t) = b3(k)$ or $rcv(j,t) = run(k)$ then there exists $s \in 1..r$ such that $srv(s) = sent(k, N_{Bjt}, N_{Bjt})$.

(5) For any $k \in Ins$, terms L and L', and name p, $snd(k) = sent(L, L')$ implies either $L = L' = p$ or neither $L = p$ nor $L' = p$.

(6) For any $k \in Ins$, $s \in 1..r$, terms L and L', and name p, $srv(s) = sent(k, L, L')$ implies either $L = L' = p$ or neither $L = p$ nor $L' = p$.

(7) Environment E is:

$$x_k : \{-\}_{K_{iS}}^{k \in Ins \text{ with } I_k = (i, j, M),\ snd(k) = sent(L, L')},$$
$$y_k : \{-\}_{K_{ABk}}^{k \in Ins \text{ with } I_k = (i, j, M),\ snd(k) = sent(L, L')},$$
$$z_s : \{-\}_{K_{Sj}}^{s \in 1..r \text{ with } I_k = (i, j, M),\ srv(s) = sent(k, L, L')}$$

(8) Substitution σ is:

$$[\{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, L\}_{K_{iS}}/x_k^{\,k \in Ins \text{ with } I_k = (i, j, M),\ snd(k) = sent(L, L')},$$
$$\{M\}_{K_{ABk}}/y_k^{\,k \in Ins \text{ with } I_k = (i, j, M),\ snd(k) = sent(L, L')},$$
$$\{S, \underline{i}, \underline{j}, K_{ABk}, L\}_{K_{Sj}}/z_s^{\,s \in 1..r \text{ with } I_k = (i, j, M),\ srv(s) = sent(k, L, L')}]$$

and substitution σ_{spec} is:

$$[\{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, L'\}_{K_{iS}}/x_k^{\,k \in Ins \text{ with } I_k = (i, j, M),\ snd(k) = sent(L, L')},$$
$$\{p_k\}_{K_{ABk}}/y_k^{\,k \in Ins \text{ with } snd(k) = sent(L, L')},$$
$$\{S, \underline{i}, \underline{j}, K_{ABk}, L'\}_{K_{Sj}}/z_s^{\,s \in 1..r \text{ with } I_k = (i, j, M),\ srv(s) = sent(k, L, L')}]$$

(9) Process R contains no free occurrence of any of the names p_k, K_{iS}, K_{Sj}, K_{ABk} and satisfies $E \vdash R$.

- Finally, we define the relation $\mathscr{S}$ as follows:

$$\mathscr{S} \triangleq \{(P^W, Q^W) \mid \text{any possible world } W\}$$

Given a possible world (snd, srv, rcv, X, E, σ, σ_{spec}, R), conditions (7) and (8) imply that E, σ, and σ_{spec} are determined by the other components of the world, and that $E \vdash \sigma$ and $E \vdash \sigma_{spec}$ hold. Moreover, σ is injective, as we show next. Let us suppose that w and w' are two variables that σ maps to the same term. Since σ maps all variables to ciphertexts under keys in one of three disjoint families, we can distinguish three possible cases:

- w is x_k and w' is $x_{k'}$ for some k, $k' \in Ins$. Since $\sigma(x_k)$ has the form $\{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, L\}_{K_{iS}}$, $\sigma(x_k)$ textually contains K_{ABk}. Similarly, $\sigma(x_{k'})$ textually contains $K_{ABk'}$ in the same position. Therefore $k = k'$, so $w = w'$.

- w is y_k and w' is $y_{k'}$ for some k, $k' \in Ins$. Since $\sigma(y_k)$ has the form $\{M\}_{K_{ABk}}$, $\sigma(y_k)$ textually contains K_{ABk}. Similarly, $\sigma(y_{k'})$ textually contains $K_{ABk'}$ in the same position. Therefore $k = k'$, so $w = w'$.

- w is z_s and w' is $z_{s'}$ for some s, $s' \in 1..r$. For some $k \in Ins$, we have $\sigma(z_s) = \{S, \underline{i}, \underline{j}, K_{ABk}, L\}_{K_{Sj}}$ where $srv(s) = sent(k, L, L')$ and $I_k = (i, j, M)$. Since $\sigma(z_{s'}) = \sigma(z_s)$, there exists L'' such that $srv(s') = sent(k, L, L'')$. By condition (3), we obtain $snd(k) = sent(N_{Ss}, N_{Ss})$ and $snd(k) = sent(N_{Ss'}, N_{Ss'})$. Therefore $s = s'$, so $w = w'$.

Thus, if σ maps two variables w and w' to the same term then $w = w'$, so σ is injective. By the same argument, σ_{spec} is injective too.

Now we consider the world $W = (snd, srv, rcv, Ins, \varnothing, \varnothing, \varnothing, R')$ where

$$R' \triangleq R \mid \prod_{k \in Ins} (\overline{c_S}\langle \underline{i} \rangle \text{ where } I_k = (i, j, M))$$

such that $snd(k) = a2$ for all $k \in Ins$, $srv(s) = s1$ for all $s \in 1..r$, and $rcv(j, t) = b1$ for all $(j, t) \in Prn \times 1..r$. The conditions for W to be possible are satisfied. In particular, X and X_2^W both equal Ins, while all other instance sets are empty. Furthermore, processes P^W and Q^W are related by $\mathscr{S}$, and are structurally equivalent to processes (14) and (15) respectively. Therefore, if we can show that $\mathscr{S} \subseteq \dot{\sim}$ it will follow that processes (14) and (15) are barbed equivalent.

To prove that $\mathscr{S} \subseteq \dot{\sim}$, we rely on Proposition 6: we show that $\mathscr{S}$ is a barbed bisimulation up to $\dot{\sim}$ and restriction. Thus, we prove, for any possible world $W = (snd, srv, rcv, X, E, \sigma, \sigma_{spec}, R)$, that: (1) any barb exhibited by P^W is also exhibited by Q^W, and vice versa, and (2) for any reaction $P^W \to P'$ there is Q' with $Q^W \to Q'$ and there is a possible world W' and names $\vec{n}$ such that $P' \dot{\sim} (\nu \vec{n})\, P^{W'}$, $Q' \dot{\sim} (\nu \vec{n})\, Q^{W'}$, and vice versa. We treat only conditions (1) and (2); the symmetric conditions can be established by a symmetric treatment.

Condition (1) holds because P^W and Q^W have almost identical structure; the only names to appear in one process but not the other are the p_k's occurring in Q^W; but the outermost restriction on the p_k's prevents their being exhibited as barbs.

To show condition (2), we first recall that P^W is:

$$\prod_{k \in Ins} A^W(k) \mid \prod_{s \in 1..r} S^W(s) \mid \prod_{(j, t) \in Prn \times 1..r} B^W(j, t) \mid R\sigma$$

and Q^W is:

$$(\nu p_k{}^{k \in Ins}) \left(\prod_{k \in Ins} A^W_{spec}(k) \mid \prod_{s \in 1..r} S^W(s) \mid \prod_{(j, t) \in Prn \times 1..r} B^W_{spec}(j, t) \mid \left(\prod_{k \in X} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_{k=}(i, j, M) \right) \mid R\sigma_{spec} \right)$$

As usual, we appeal to Proposition 3 in order to analyze the reactions of P^W in terms of its possible commitments. Processes $A^W(k)$, $S^W(s)$, $B^W(j, t)$ have only input or τ commitments, whereas the arbitrary process $R\sigma$ may have input, output, or τ commitments. Therefore, a reaction of P^W can arise in only one of the following ways:

(A) from the interaction of an output commitment $R\sigma \xrightarrow{\alpha} (\nu \vec{n})\langle L_1 \rangle\, R_1$ and an input commitment of one of the following seven kinds of process:

- $A^W(k) = A2'(I_k, K_{ABk})$
  where $k \in Ins$ and $snd(k) = a2$,
- $S^W(s) = S1'(N_{Ss})$
  where $s \in 1..r$ and $srv(s) = s1$,

- $S^W(s) = S2(i, N_{Ss})$
  where $s \in 1..r$ and $srv(s) = s2(i)$,
- $S^W(s) = S4(i, j, K_{ABk})$
  where $s \in 1..r$, $srv(s) = s4(k)$, and $I_k = (i, j, M)$,
- $B^W(j, t) = B\,1'(j, \tau\,.\,F, N_{Bjt})$
  where $(j, t) \in Prn \times 1..r$ and $rcv(j, t) = b1$,
- $B^W(j, t) = B2(j, \tau.F, N_{Bjt})$
  where $(j, t) \in Prn \times 1..r$ and $rcv(j, t) = b2$,
- $B^W(j, t) = B3(i, j, \tau.F, K_{ABk})$
  where $(j, t) \in Prn \times 1..r$, $rcv(j, t) = b3(k)$, and $I_k = (i, j, M)$,

(B) from a τ commitment $B^W(j, t) = \tau.F(\underline{i}, \underline{j}, M) \xrightarrow{\tau} F(\underline{i}, \underline{j}, M)$ where $rcv(j, t) = run(k)$ and $I_k = (i, j, M)$,

(C) from a τ commitment $R\sigma \xrightarrow{\tau} R_1$.

In case (A), we may assume that the bound names $\vec{n}$ are fresh. Since W is possible, it follows that $E \vdash \sigma$ and $E \vdash \sigma_{spec}$, and that both σ and σ are injective substitutions. Therefore, given the commitment $R\sigma \xrightarrow{\alpha} (v\vec{n})\langle L_1 \rangle\, R_1$, Lemma 9(2) guarantees that there is an agent A such that $E \vdash A$, $fv(A) \subseteq fv(R)$, $fn(A) \subseteq fn(R)$, and $(v\vec{n})\langle L_1 \rangle\, R_1 = A\sigma$, and moreover that $R\sigma_{spec} \xrightarrow{\alpha} A\sigma_{spec}$. From $(v\vec{n})\langle L_1 \rangle\, R_1 = A\sigma$ it follows there are L_2 and R_2 such that $A = (v\vec{n})\langle L_2 \rangle\, R_2$, $L_1 = L_2\sigma$, and $R_1 = R_2\sigma$.

We now examine the input commitments of the seven kinds of process above (ordered according to the enumeration of messages in the informal description of the protocol, rather than as in the list above) and exhibit in each case a possible world W' such that $P' \nsim (v\vec{n})\, P^{W'}$ and there is Q' with $Q^W \to Q'$ and $Q' \nsim (v\vec{n})\, Q^{W'}$, where $\vec{n}$ are the names generated in the commitment of R.

(1) The reaction $P^W \to P'$, where

$$P' \equiv (v\vec{n}) \Bigg(\prod_{k \in Ins} A^W(k) \,\Big|\, \prod_{s' \in 1..r - \{s\}} S^W(s') \,\Big|\, \prod_{i \in Prn} [L_2\sigma \text{ is } i](\overline{c_i}\langle N_{Ss} \rangle \,|\, S2(i, N_{Ss})) \,\Big|\, \prod_{(j,t) \in Prn \times 1..r} B^W(j, t) \,\Big|\, R_2\sigma \Bigg)$$

arises when α is $\overline{c_S}$, and there is an input commitment

$$S1'(N_{Ss}) \xrightarrow{c_S} (x) \prod_{i \in Prn} [x \text{ is } i](\overline{c_i}\langle N_{Ss} \rangle \mid S2(i, N_{Ss}))$$

for some $s \in 1..r$ such that $srv(s) = s1$.

We argue by cases on whether there is $i \in Prn$ such that $L_2\sigma = \underline{i}$.

When there is $i \in Prn$ such that $L_2\sigma = \underline{i}$, we can simplify P' as follows:

$$P' \equiv (\nu\vec{n}) \left(\prod_{k \in Ins} A^W(k) \;\middle|\; \prod_{s' \in 1..r-\{s\}} S^W(s') \;\middle|\; S2(i, N_{Ss}) \right.$$
$$\left. \prod_{(j,t) \in Prn \times 1..r} B^W(j, t) \;\middle|\; \overline{c_i}\langle N_{Ss} \rangle \;\middle|\; R_2\sigma \right)$$

We set

$$W' = (snd, srv', rcv, X, E, \sigma, \sigma_{spec}, \overline{c_i}\langle N_{Ss} \rangle \mid R_2)$$

where srv' is identical to srv except that $srv'(s) = s2(1)$. With this definition, $P' \equiv (\nu\vec{n})\, P^{W'}$. Given the form of σ, $L_2\sigma = \underline{i}$ implies that $L_2 = \underline{i}$, and therefore also that $L_2\sigma_{spec} = \underline{i}$. Therefore, $Q^W \to (\nu\vec{n})\, Q^{W'}$, so we let $Q' = (\nu\vec{n})\, Q^{W'}$.

It remains to prove that the world W' is possible. Conditions (1) and (2), which are about the instance sets of W', must hold since the instance sets of W' equal those of W, which itself is possible. Conditions (3) and (6) concern servers in states $s4$ and $sent(k, L, L')$; they hold for W and continue to hold for W' as no servers have entered those states. Conditions (4) and (5) continue to hold in W' as no senders or receivers have changed state. Conditions (7) and (8) concerning E, σ, and σ_{spec} hold, since W is possible, no senders have entered or left a $sent(L, L')$ state, and no servers have entered or left a $sent(k, L, L')$ state. Finally, condition (9) is that $\overline{c_i}\langle N_{Ss} \rangle \mid R_2$ contains no free occurrence of any of the names p_k, K_{iS}, K_{Sj}, K_{ABk} and that $E \vdash \overline{c_i}\langle N_{Ss} \rangle \mid R_2$. It holds since the same condition holds for R, and we know that $fn((\nu\vec{n})\langle L_2 \rangle\, R_2) \subseteq fn(R)$, that the names $\vec{n}$ are fresh, and that $E \vdash (\nu\vec{n})\langle L_2 \rangle\, R_2$. Therefore, W' is a possible world.

Otherwise, when there is no $i \in Prn$ such that $L_2\sigma = \underline{i}$, we can simplify P' as follows:

$$P' \;\tilde{\cdot}\; (\nu\vec{n}) \left(\prod_{k \in Ins} A^W(k) \;\middle|\; \prod_{s' \in 1..r-\{s\}} S^W(s') \;\middle|\; \right.$$
$$\left. \prod_{(j,t) \in Prn \times 1..r} B^W(j, t) \;\middle|\; R_2\sigma \right)$$

We set:

$$W' = (snd, srv', rcv, X, E, \sigma, \sigma_{spec}, R_2)$$

where srv' is identical to srv except that $srv'(s) = stuck$. With this definition, $P' \;\tilde{\cdot}\; (\nu\vec{n})\, P^{W'}$. Given the form of σ and σ_{spec}, $L_2\sigma \neq \underline{i}$ implies that $L_2\sigma_{spec} \neq \underline{i}$ for every $i \in Prn$. Letting

$$Q' \triangleq (v\vec{n})(vp_k{}^{k \in Ins})\Big(\prod_{k \in Ins} A^W_{spec}(k) \Big| \prod_{s' \in 1..r-\{s\}} S^W(s') \Big|$$

$$\prod_{i \in Prn} [L_2\sigma_{spec}\ is\ \underline{i}](\overline{c_i}\langle N_{Ss}\rangle \mid S2(i, N_{Ss})) \Big|$$

$$\prod_{(j,t) \in Prn \times 1..r} B^W_{spec}(j, t) \Big|$$

$$\Big(\prod_{k \in X} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_{k=}(i, j, M)\Big) \Big|$$

$$R_2\sigma_{spec}\Big)$$

we obtain $Q^W \to Q' \backsim (v\vec{n})\, Q^{W'}$.

In this case, it remains to show that the world W' is possible. Conditions (1) and (2) concern the instance sets of W'. We have:

$$X_3^{W'} = \{k \in X_3^W \mid snd(k) \neq sent(N_{Ss}, N_{Ss})\}$$

from which it follows that $X_3^{W'} \subseteq X_3^W$. All the other instance sets of W' equal those of W. Since conditions (1) and (2) hold for W, they hold also for W'. The rest of the proof that the world W' is possible is as in the case where there is $i \in Prn$ such that $L_2\sigma = \underline{i}$.

(2) The reaction $P^W \to P'$, where

$$P' \equiv (v\vec{n})\Big(\prod_{k' \in Ins-\{k\}} A^W(k') \Big|$$

$$\overline{c_S}\langle(\underline{i}, \{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, L_2\sigma\}\ K_{iS})\rangle \mid \overline{c_j}\langle(\underline{i}, \{M\}_{K_{ABk}})\rangle \Big|$$

$$\prod_{s \in 1..r} S^W(s) \Big| \prod_{(j,t) \in Prn \times 1..r} B^W(j, t) \Big| R_2\sigma\Big)$$

arises when α is $\overline{c_i}$, and there is an input commitment

$$A2'(I_k, K_{ABk}) \xrightarrow{c_i} (x)(\overline{c_S}\langle(\underline{i}\{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, x\}_{K_{iS}})\rangle \mid \overline{c_j}\langle(\underline{i}, \{M\}_{K_{ABk}})\rangle),$$

for some $k \in Ins$ such that $snd(k) = a2$ and $I_k = (i, j, M)$. We set:

$$W' = (snd', srv, rcv, X, E', \sigma', \sigma'_{spec}, R_2 \mid \overline{c_S}\langle x_k\rangle \mid \overline{c_j}\langle(\underline{i}, y_k)\rangle)$$

where snd' is identical to snd except that $snd'(k) = sent(L_2\sigma, L_2\sigma_{spec})$, and

$$E' \triangleq E, x_k: \{-\}_{K_{iS}}, y_k: \{-\}_{K_{ABk}}$$
$$\sigma' \triangleq \sigma, \{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, L_2\sigma\}_{K_{iS}}/x_k, \{M\}_{K_{ABk}}/y_k$$
$$\sigma'_{spec} \triangleq \sigma_{spec}, \{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, L_2\sigma_{spec}\}_{K_{iS}}/x_k, \{M\}_{K_{ABk}}/y_k$$

With this definition, $P' \equiv (\nu\vec{n})\, P^{W'}$; moreover, $Q^W \to (\nu\vec{n})\, Q^{W'}$.

It remains to show that the world W' is possible. First, we consider the instance sets of W'. They are equal to those of W, except for:

$$X_2^{W'} = X_2^W - \{k\} \qquad \text{while} \quad k \in X_2^W$$
$$X_3^{W'} = \begin{cases} X_3^W \cup \{k\} & \text{if } \exists s \in 1..r,\ i' \in Prn \\ & \quad (L_2\sigma = L_2\sigma_{spec} = N_{Ss}\ \&\ \\ & \quad srv(s) \in \{s1, s2(i')\}) \\ X_3^W & \text{otherwise} \end{cases}$$

Therefore, since conditions (1) and (2) hold for W, they hold also for W'. Condition (5) holds for W' because there are no names in the range of σ or σ_{spec}, so for any name n either $L_2\sigma = L_2\sigma_{spec} = n$ or neither $L_2\sigma = n$ nor $L_2\sigma_{spec} = n$. Conditions (3), (4), (6), (7), (8), and (9) hold for W, and it follows easily that they continue to hold for W'.

(3) The reaction $P^W \to P'$, where

$$P' \equiv (\nu\vec{n}) \Big(\prod_{k \in Ins} A^W(k) \,\Big|\, \prod_{s' \in 1..r - \{s\}} S^W(s') \,\Big|\, \prod_{(j,t) \in Prn \times 1..r} B^W(j,t) \,\Big|\, R_2\sigma \,\Big|$$
$$let\ (y_1, y_2) = L_2\sigma\ in$$
$$[y_1\ is\ i]\ case\ y_2\ of\ \{z_1, z_2, z_3, z_4, z_5\}_{K_{iS}}\ in$$
$$\prod_{j \in Prn} [z_1\ is\ \underline{i}][z_2\ is\ \underline{i}][z_3\ is\ \underline{j}][z_5\ is\ N_{Ss}]$$
$$(\overline{c_j}\langle * \rangle \mid S4(i, j, z_4))\Big)$$

arises when α is $\overline{c_S}$, and there is an input commitment

$$S2(i, N_{Ss}) \xrightarrow{c_S} (x) let(y_1, y_2) = x\ in$$
$$[y_1\ is\ \underline{i}]\ case\ y_2\ of\ \{z_1, z_2, z_3, z_4, z_5\}_{K_{iS}}\ in$$
$$\prod_{j \in Prn} [z_1\ is\ \underline{i}][z_2\ is\ \underline{i}][z_3\ is\ \underline{j}][z_5\ is\ N_{Ss}]$$
$$(\overline{c_j}\langle * \rangle \mid S4(i, j, z_4))$$

for some $s \in 1..r$ with $srv(s) = s2(i)$.

We argue by cases on whether $L_2\sigma$ is a pair with first component $\underline{i}$ and second component a ciphertext under K_{iS} containing N_{Ss} as last field. By condition (8), $L_2\sigma$ has $\underline{i}$ as first component if and only if L_2 has $\underline{i}$ as first component. Similarly, since $fn(L_2) \subseteq fn(R) \cup \{\vec{n}\}$, the second component of $L_2\sigma$ is a ciphertext under K_{iS} containing N_{Ss} if and only if the second component of L_2 is a variable x_k for some $k \in Ins$ such that $snd(k) = sent(N_{Ss}, L')$ for some L'. In this case, the second component of $L_2\sigma$ is $\{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, N_{Ss}\}_{K_{iS}}$ where $I_k = (i, j, M)$. Thus, $L_2\sigma$ determines k uniquely because of the presence of K_{ABk}. By condition (5), $L' = N_{Ss}$ and $snd(k) = sent(N_{Ss}, N_{Ss})$, so if L_2 has the form $(\underline{i}, x_k)$, then $L_2\sigma$ and $L_2\sigma_{spec}$ both equal $(\underline{i}, \{\underline{i}, \underline{i}, \underline{j}, K_{ABk}, N_{Ss}\}_{K_{iS}})$. Conversely, the form of $L_2\sigma_{spec}$ determines the form of $L_2\sigma$.

Assuming that $L_2\sigma$ is a pair of the form described, we can simplify P' as follows:

$$P' \equiv (\nu\vec{n}) \left(\prod_{k \in Ins} A^W(k) \;\middle|\; \prod_{s' \in 1..r - \{s\}} S^W(s') \;\middle|\; \prod_{(j,t) \in Prn \times 1..r} B^W(j,t) \;\middle|\; S4(i, j, K_{ABk}) \mid R_2\sigma \mid \overline{c_j}\langle * \rangle \right)$$

where i, j, and k are defined as explained above. We set:

$$W' = (snd, srv', rcv, X, E, \sigma, \sigma_{spec}, R_2 \mid \overline{c_j}\langle * \rangle)$$

where srv' is identical to srv except that $srv'(s) = s4(k)$. With this definition, $P' \equiv (\nu\vec{n})\, P^{W'}$ and $Q^W \to (\nu\vec{n})\, Q^{W'}$.

It remains to show that the world W' is possible. All the instance sets of W' equal those of W, except for:

$$X_3^{W'} = X_3^W - \{k' \in X_3^W \mid snd(k') = sent(N_{Ss}, N_{Ss})\}$$
$$\text{while} \quad k \in X_3^W$$
$$X_5^{W'} = X_5^W \cup \{k\}$$

In particular, $k \notin X_3^{W'}$. Therefore, conditions (1) and (2) hold for W'. Conditions (3), (4), (5), (6), (7), (8), and (9) hold for W, and it follows easily that they continue to hold for W'. For condition (3), we use the fact that $snd(k) = sent(N_{Ss}, N_{Ss})$.

On the other hand, if $L_2\sigma$ is not of the form described, we can simplify P' as follows:

$$P' \;\tilde{\cdot}\; (\nu\vec{n}) \left(\prod_{k \in Ins} A^W(k) \;\middle|\; \prod_{s' \in 1..r - \{s\}} S^W(s') \;\middle|\; \prod_{(j,t) \in Prn \times 1..r} B^W(j,t) \;\middle|\; R_2\sigma \right)$$

We set

$$W' = (snd, srv', rcv, X, E, \sigma, \sigma_{spec}, R_2)$$

where srv' is identical to srv except that $srv'(s) = stuck$. With this definition, $P' \stackrel{\cdot}{\sim} (v\vec{n})\, P^{W'}$. Letting

$$\begin{aligned} Q' \triangleq (v\vec{n})(vp_k{}^{k \in Ins}) \Big(& \prod_{k \in Ins} A^W_{spec}(k) \;\Big|\; \prod_{s' \in 1..r - \{s\}} S^W(s') \;\Big| \\ & let\ (y_1, y_2) = L_2 \sigma_{spec}\ in \ldots \;\Big| \\ & \prod_{(j,t) \in Prn \times 1..r} B^W_{spec}(j, t) \;\Big| \\ & \Big(\prod_{k \in X} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_{k=}(i, j, M) \Big) \;\Big| \\ & R_2 \sigma_{spec} \Big) \end{aligned}$$

where the omitted code gets stuck, we obtain $Q^W \to Q' \stackrel{\cdot}{\sim} (v\vec{n})\, Q^{W'}$.

In this case, it is easy to check that the world W' is possible. All the instance sets of W' equal those of W, except for:

$$X_3^{W'} = \{k \in X_3^W \mid snd(k) \neq sent(N_{Ss}, N_{Ss})\}$$

so $X_3^{W'} \subseteq X_3^W$.

(4) The reaction $P^W \to P'$, where

$$\begin{aligned} P' \equiv (v\vec{n}) \Big(& \prod_{k \in Ins} A^W(k) \;\Big|\; \prod_{s \in 1..r} S^W(s) \;\Big| \\ & \prod_{(j',t') \in Prn \times 1..r - \{(j,t)\}} B^W(j', t') \;\Big| \\ & \overline{c_S}\langle N_{Bjt} \rangle \;\Big|\; B2(j, \tau.F, N_{Bjt}) \;\Big|\; R_2 \sigma \Big) \end{aligned}$$

arises when α is $\overline{c_j}$ and there is an input commitment

$$B1'(j, \tau.F, N_{Bjt}) \xrightarrow{c_j} (x)(\overline{c_S}\langle N_{Bjt} \rangle \mid B2(j, \tau.F, N_{Bjt}))$$

for some $(j, t) \in Prn \times 1..r$ such that $rcv(j, t) = b1$. We set:

$$W' = (snd, srv, rcv', X, E, \sigma, \sigma_{spec}, R_2 \mid \overline{c_S}\langle N_{Bjt} \rangle)$$

where rcv' is identical to rcv except that $rcv'(j, t) = b2$. With this definition, $P' \equiv (v\vec{n})\, P^{W'}$ and $Q^W \to (v\vec{n})\, Q^{W'}$. Given that W is a possible world, so is W'; in particular, the instance sets of W' equal those of W.

(5) The reaction $P^W \to P'$, where

$$P' \equiv (v\vec{n}) \left(\prod_{k \in Ins} A^W(k) \;\middle|\; \prod_{s' \in 1..r - \{s\}} S^W(s') \;\middle|\; \prod_{(j,t) \in Prn \times 1..r} B^W(j,t) \;\middle|\; \overline{c_j}\langle \{S, \underline{i}, \underline{j}, K_{ABk}, L_2\sigma\}_{K_{Sj}} \rangle \;\middle|\; R_2\sigma \right)$$

arises when α is $\overline{c_S}$, and there is an input commitment

$$S4(i, j, K_{ABk}) \xrightarrow{c_S} (x)\, \overline{c_j}\langle \{S, \underline{i}, \underline{j}, K_{ABk}, x\}_{K_{Sj}} \rangle$$

for some $s \in 1..r$ such that $srv(s) = s4(k)$ and $I_k = (i, j, M)$.

We set:

$$W' = (snd, srv', rcv, X, E', \sigma', \sigma'_{spec}, R_2 \mid \overline{c_j}\langle z_s \rangle)$$

where srv' is identical to srv except that $srv'(s) = sent(k, L_2\sigma, L_2\sigma_{spec})$, and:

$$E' \triangleq E, z_s : \{-\}_{K_{Sj}}$$
$$\sigma' \triangleq \sigma, \{S, \underline{i}, \underline{j}, K_{ABk}, L_2\sigma\}_{K_{Sj}}/z_s$$
$$\sigma'_{spec} \triangleq \sigma_{spec}, \{S, \underline{i}, \underline{j}, K_{ABk}, L_2\sigma_{spec}\}_{K_{Sj}}/z_s$$

With this definition, $P' \equiv (v\vec{n})\, P^{W'}$ and $Q^W \to (v\vec{n})\, Q^{W'}$.

It remains to show that the world W' is possible. First, we note that if $srv(s') = s4(k)$ then $s = s'$, because $srv(s) = s4(k)$ and by condition (3). Therefore, all the instance sets of W' equal those of W, except for:

$$X_5^{W'} = X_5^W - \{k\}$$
$$X_6^{W'} = \begin{cases} X_6^W \cup \{k\} & \text{sometimes—when is unimportant} \\ X_6^W & \text{otherwise} \end{cases}$$

So conditions (1) and (2) hold for W'. Since W satisfies conditions (4) and (5), so does W', trivially. Condition (3) for W implies that $snd(k) = sent(N_{Ss}, N_{Ss})$; it follows that condition (3) holds for W'. Condition (6) holds for W' because there can be no names in the ranges of σ and σ_{spec}, so, for any name n, either $L_2\sigma = L_2\sigma_{spec} = n$ or neither $L_2\sigma = n$ nor $L_2\sigma_{spec} = n$. Conditions (7), (8), and (9) for W' are easy consequences of the corresponding conditions for W.

(6) The reaction $P^W \to P'$, where

$$P' \equiv (\nu \vec{n}) \Big(\prod_{k \in Ins} A^W(k) \Big| \prod_{s \in 1..r} S^W(s) \Big| \prod_{(j', t') \in Prn \times 1..r - \{(j,t)\}} B^W(j', t') \Big| R_2\sigma \Big| \textit{case } L_2\sigma \textit{ of } \{y_1, y_2, y_3, y_4, y_5\}_{K_{Sj}} \textit{ in } \prod_{i \in Prn} [y_1 \textit{ is } S][y_2 \textit{ is } \underline{i}][y_3 \textit{ is } \underline{j}][y_5 \textit{ is } N_{Bjt}] B3(i, j, \tau.F, y_4) \Big)$$

arises when α is $\overline{c_j}$, and there is an input commitment

$$B2(j, \tau.F, N_{Bjt}) \xrightarrow{c_j} (x) \textit{ case } x \textit{ of } \{y_1, y_2, y_3, y_4, y_5\}_{K_{Sj}} \textit{ in } \prod_{i \in Prn} [y_1 \textit{ is } S][y_2 \textit{ is } \underline{i}][y_3 \textit{ is } \underline{j}][y_5 \textit{ is } N_{Bjt}] B3(i, j, \tau.F, y_4)$$

for some $(j, t) \in Prn \times 1..r$ with $rcv(j, t) = b2$.

We argue by cases on whether $L_2\sigma$ is a ciphertext under K_{Sj} containing N_{Bjt} as last field. By condition (8), since $fn(L_2) \subseteq fn(R) \cup \{\vec{n}\}$, $L_2\sigma$ is a ciphertext under K_{Sj} containing N_{Bjt} if and only if L_2 is a variable z_s for some $s \in 1..r$ such that $srv(s) = sent(k, N_{Bjt}, L')$ for some k and L'. In this case, $L_2\sigma$ is $\{S, \underline{i}, \underline{j}, K_{ABk}, N_{Bjt}\}_{K_{Sj}}$ where $I_k = (i, j, M)$. Thus, $L_2\sigma$ determines k uniquely because of the presence of K_{ABk}. By condition (6), $L' = N_{Bjt}$ and $srv(s) = sent(k, N_{Bjt}, N_{Bjt})$, so if L_2 is z_s then $L_2\sigma$ and $L_2\sigma_{spec}$ both equal $\{S, \underline{i}, \underline{j}, K_{ABk}, N_{Bjt}\}_{K_{Sj}}$. Conversely, the form of $L_2\sigma_{spec}$ determines the form of $L_2\sigma$.

Assuming that $L_2\sigma$ is of the form described, we can simplify P' as follows:

$$P' \equiv (\nu \vec{n}) \Big(\prod_{k \in Ins} A^W(k) \Big| \prod_{s \in 1..r} S^W(s) \Big| \prod_{(j', t') \in Prn \times 1..r - \{(j,t)\}} B^W(j', t') \Big| B3(i, j, \tau.F, K_{ABk}) \Big| R_2\sigma \Big)$$

We set:

$$W' = (snd, srv, rcv', X, E, \sigma, \sigma_{spec}, R_2)$$

where rcv' is identical to rcv except that $rcv'(j, t) = b3(k)$. With this definition, $P' \equiv (\nu \vec{n})\, P^{W'}$ and $Q^W \to (\nu \vec{n})\, Q^{W'}$.

It remains to check that the world W' is possible. All the instance sets of W' equal those of W, except for:

$$X_6^{W'} = X_6^W - \{k' \text{ where } I_{k'} = (i', j, M') | \exists s' \in 1..r(srv(s') = sent(k', N_{Bjt}, N_{Bjt}))\}$$
$$\text{while} \quad k \in X_6^W$$
$$X_7^{W'} = X_7^W \cup \{k\}$$

In particular, $k \notin X_6^{W'}$. Therefore, conditions (1) and (2) hold for W'. Conditions (3), (5), (6), (7), (8), and (9) hold for W, and it follows easily that they continue to hold for W'. Condition (4) holds for W' because $srv(s) = sent(k, N_{Bjt}, N_{Bjt})$.

On the other hand, if $L_2\sigma$ is not of the form described, we can simplify P' as follows:

$$P' \stackrel{\cdot}{\sim} (v\vec{n})\left(\prod_{k \in Ins} A^W(k) \mid \prod_{s \in 1..r} S^W(s) \mid \prod_{(j', t') \in Prn \times 1..r - \{(j,t)\}} B^W(j', t') \mid R_2\sigma\right)$$

We set:

$$W' = (snd, srv, rcv', X, E, \sigma, \sigma_{spec}, R_2)$$

where rcv' is identical to rcv except that $rcv'(j, t) = stuck$. With this definition, $P' \stackrel{\cdot}{\sim} (v\vec{n})\, P^{W'}$. Letting

$$Q' \triangleq (v\vec{n})\,(vp_k{}^{k \in Ins})\left(\prod_{k \in Ins} A^W_{spec}(k) \mid \prod_{s \in 1..r} S^W(s) \mid \prod_{(j', t') \in Prn \times 1..r - \{(j,t)\}} B^W_{spec}(j', t') \mid \right.$$
$$case\ L_2\sigma\ of\ \{y_1, y_2, y_3, y_4, y_5\}_{K_{Sj}}\ in\ \ldots \mid$$
$$\left.\left(\prod_{k \in X} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_k = (i, j, M)\right) \mid R_2\sigma_{spec}\right)$$

where the omitted code gets stuck, we obtain $Q^W \to Q' \stackrel{\cdot}{\sim} (v\vec{n})\, Q^{W'}$.

In this case, it is easy to check that the world W' is possible. All the instance sets of W' equal those of W, except for:

$$X_6^{W'} = X_6^W - \{k \text{ where } I_k = (i, j, M) | \exists s \in 1..r(srv(s) = sent(k, N_{Bjt}, N_{Bjt}))\}$$

so $X_6^{W'} \subseteq X_6^W$.

(7) The reaction $P^W \to P'$, where

$$P' \equiv (\nu \vec{n}) \left(\prod_{k \in Ins} A^W(k) \,\middle|\, \prod_{s \in 1..r} S^W(s) \,\middle|\, \prod_{(j', t') \in Prn \times 1..r - \{(j, t)\}} B^W(j', t') \,\middle|\, R_2\sigma \,\middle|\, let\ (y_1, y_2) = L_2\sigma\ in\ [y_1\ is\ \underline{i}]\ case\ y_2\ of\ \{z\}_{K_{ABk}}\ in\ \tau.F(\underline{i}, \underline{j}, z) \right)$$

arises when α is $\overline{c_j}$, and there is an input commitment

$$B3(i, j, \tau.F, K_{ABk}) \xrightarrow{c_j} (x)\ let\ (y_1, y_2) = x\ in\ [y_1\ is\ \underline{i}]case\ y_2\ of\ \{z\}_{K_{ABk}}\ in\ \tau.F(\underline{i}, \underline{j}, z)$$

for some $k \in Ins$ and $(j, t) \in Prn \times 1..r$ such that $rcv(j, t) = b3(k)$ and $I_k = (i, j, M)$ for some M.

We argue by cases on whether $L_2\sigma$ is a pair with first component $\underline{i}$ and second component a ciphertext under K_{ABk}. By condition (8), $L_2\sigma$ has $\underline{i}$ as first component if and only if L_2 has $\underline{i}$ as first component. Similarly, since $fn(L_2) \subseteq fn(R) \cup \{\vec{n}\}$, the second component of $L_2\sigma$ is a ciphertext under K_{ABk} if and only if the second component of L_2 is y_k and $snd(k) = sent(L, L')$ for some L and L'. In this case, the second component of $L_2\sigma$ is $\{M\}_{K_{ABk}}$. Thus, if L_2 has the form $(\underline{i}, y_k)$, then $L_2\sigma$ equals $(\underline{i}, \{M\}_{K_{ABk}})$, while $L_2\sigma_{spec}$ equals $(\underline{i}, \{p_k\}_{K_{ABk}})$. Conversely, the form of $L_2\sigma_{spec}$ determines the form of $L_2\sigma$.

Assuming that $L_2\sigma$ is a pair of the form described, we can simplify P' as follows:

$$P' \equiv (\nu \vec{n}) \left(\prod_{k \in Ins} A^W(k) \,\middle|\, \prod_{s \in 1..r} S^W(s) \,\middle|\, \prod_{(j', t') \in Prn \times 1..r - \{(j, t)\}} B^W(j', t') \,\middle|\, R_2\sigma \,\middle|\, \tau.F(\underline{i}, \underline{j}, M) \right)$$

We set

$$W' = (snd, srv, rcv', X, E, \sigma, \sigma_{spec}, R_2)$$

where rcv' is identical to rcv except that $rcv'(j, t) = run(k)$. With this definition, $P' \equiv (\nu\vec{n})\, P^{W'}$ and $Q^W \to (\nu\vec{n})\, Q^{W'}$.

In order to check that the world W' is possible, we first consider the instance sets of W'. First, we argue that $k \notin X_7^{W'}$. It suffices to show that if $rcv(j, t') = b3(k)$ then in fact $t = t'$. Condition (4) for W says that there exists $s \in 1..r$ such that $srv(s) = sent(k, N_{Bjt}, N_{Bjt})$, and that if $rcv(j, t') = b3(k)$ then there exists $s' \in 1..r$ such that $srv(s') = sent(k, N_{Bjt'}, N_{Bjt'})$. Condition (3) for W says that $snd(k) = sent(N_{Ss}, N_{Ss})$ and $snd(k) = sent(N_{Ss'}, N_{Ss'})$. Therefore, $s = s'$ and then $t = t'$. We conclude that $k \notin X_7^{W'}$. We obtain that the instance sets of W' equal those of W except for

$$X_7^{W'} = X_7^W - \{k\} \qquad \text{while} \quad k \in X_7^W$$
$$X_8^{W'} = X_8^W \cup \{k\}$$

So conditions (1) and (2) hold for W'. Conditions (3), (4), (5), (6), (7), (8), and (9) hold for W, and it follows easily that they continue to hold for W'.

On the other hand, if $L_2\sigma$ is not of the form described, we can simplify P' as follows:

$$P' \;\tilde{\sim}\; (\nu\vec{n}) \left(\prod_{k \in Ins} A^W(k) \;\middle|\; \prod_{s \in 1..r} S^W(s) \;\middle|\; \prod_{(j', t') \in Prn \times 1..r - \{(j,t)\}} B^W(j', t') \;\middle|\; R_2\sigma \right)$$

We set:

$$W' = (snd, srv, rcv', X, E, \sigma, \sigma_{spec}, R_2)$$

where rcv' is identical to rcv except that $rcv'(j, t) = stuck$. With this definition, $P' \;\tilde{\sim}\; (\nu\vec{n})\, P^{W'}$. Letting

$$\begin{aligned} Q' \triangleq (\nu\vec{n})\, (\nu p_k{}^{k \in Ins}) \Bigg(& \prod_{k \in Ins} A^W_{spec}(k) \;\Big|\; \prod_{s \in 1..r} S^W(s) \;\Big| \\ & \prod_{(j', t') \in Prn \times 1..r - \{(j,t)\}} B^W_{spec}(j', t') \;\Big| \\ & let\ (y_1, y_2) L_2\sigma\ in \ldots \;\Big| \\ & \left(\prod_{k \in X} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_k = (i, j, M) \right) \Big| \\ & R_2\sigma_{spec} \Bigg) \end{aligned}$$

where the omitted code gets stuck, we obtain $Q^W \to Q' \;\tilde{\sim}\; (\nu\vec{n})\, Q^{W'}$.

The proof that W' is possible is almost identical to that just given for the other case; the only change is that $X_8^{W'} = X_8^W$.

This completes case (A).

In case (B), the reaction $P^W \to P'$, where

$$P' \equiv \prod_{k \in Ins} A^W(k) \Big| \prod_{s \in 1..r} S^W(s) \Big| \prod_{(j', t') \in Prn \times 1..r - \{(j, t)\}} B^W(j', t') \Big| F(\underline{i}, \underline{j}, M) \Big| R\sigma$$

arises from the τ commitment

$$B^W(j, t) \xrightarrow{\tau} F(\underline{i}, \underline{j}, M)$$

for some $(j, t) \in Prn \times 1..r$ such that $rcv(j, t) = run(k)$ and $I_k = (i, j, M)$ for some $k \in Ins$. Note that $k \in X$, since $rcv(j, t) = run(k)$ implies $k \in X_8^W \subseteq X$. We set:

$$W' = (snd, srv, rcv', X', E, \sigma, \sigma_{spec}, F(\underline{i}, \underline{j}, M) \mid R),$$

where rcv' is identical to rcv except that $rcv'(j, t) = done$ and where $X' = X - \{k\}$. With this definition, $P' \equiv P^{W'}$. Moreover, we have:

$$\begin{aligned} Q^W = (\nu p_k{}^{k \in Ins}) \Big(& \prod_{k \in Ins} A^W(k) \Big| \prod_{s \in 1..r} S^W(s) \Big| \\ & \prod_{(j, t) \in Prn \times 1..r} B^W_{spec}(j, t) \Big| \\ & \Big(\prod_{k \in X} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_k = (i, j, M) \Big) \Big| R\sigma_{spec} \Big) \\ \equiv (\nu p_k{}^{k \in Ins}) \Big(& \prod_{k \in Ins} A^W(k) \Big| \prod_{s \in 1..r} S^W(s) \Big| \\ & \prod_{(j', t') \in Prn \times 1..r - \{(j, t)\}} B^W_{spec}(j', t') \Big| \\ & \Big(\prod_{k \in X'} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_k = (i, j, M) \Big) \Big| \\ & \overline{p_k}\langle * \rangle \Big| p_k(x).F(\underline{i}, \underline{j}, M) \Big| R\sigma_{spec} \Big) \end{aligned}$$

$$\rightarrow (vp_k{}^{k \in Ins}) \Bigg(\prod_{k \in Ins} A^W(k) \Bigg| \prod_{s \in 1..r} S^W(s) \Bigg|$$

$$\prod_{(j', t') \in Prn \times 1..r - \{(j, t)\}} B^W_{spec}(j', t') \Bigg|$$

$$\Bigg(\prod_{k \in X'} p_k(x).F(\underline{i}, \underline{j}, M) \text{ where } I_k = (i, j, M) \Bigg) \Bigg|$$

$$F(\underline{i}, \underline{j}, M) \Bigg| R\sigma_{spec} \Bigg)$$

$$\equiv Q^{W'}$$

In order to check that the world W' is possible, we first argue that $k \notin X_8^{W'}$. It suffices to show that if $rcv(j, t') = run(k)$ then in fact $t = t'$. Condition (4) for W says that there exists $s \in 1..r$ such that $srv(s) = sent(k, N_{Bjt}, N_{Bjt})$, and that if $rcv(j, t') = run(k)$ then there exists $s' \in 1..r$ such that $srv(s') = sent(k, N_{Bjt'}, N_{Bjt'})$. Condition (3) for W says that $snd(k) = sent(N_{Ss}, N_{Ss})$ and $snd(k) = sent(N_{Ss'}, N_{Ss'})$. Therefore, $s = s'$ and then $t = t'$. We conclude that $k \notin X_8^{W'}$. We obtain that the instance sets of W' equal those of W except for:

$$X_8^{W'} = X_8^W - \{k\}$$

So conditions (1) and (2) hold for W'. Conditions (3), (4), (5), (6), (7), (8), and (9) hold for W, and it follows easily that they continue to hold for W'. For condition (9), we rely on the fact that $F(\underline{i}, \underline{j}, M)$ is a closed process and that it cannot contain free occurrences of any of the names p_k, K_{iS}, K_{Sj}, K_{ABk}. (The abstraction F cannot contain free occurrences of those names because of our general convention that bound parameters of the protocol do not occur free in F. The term M cannot because it is part of the arguments to Sys and Sys_{spec}.)

Finally, in case (C), the reaction $P^W \rightarrow P'$, where

$$P' \equiv \prod_{k \in Ins} A^W(k) \Bigg| \prod_{s \in 1..r} S^W(s) \Bigg| \prod_{(j, t) \in Prn \times 1..r} B^W(j, t) \Bigg| R_1$$

arises from the τ commitment $R\sigma \xrightarrow{\tau} R_1$. Lemma 9(2) implies that there is a process R_2 such that $E \vdash R_2$, $fv(R_2) \subseteq fv(R)$, $fn(R_2) \subseteq fn(R)$, $R_1 = R_2\sigma$, and $R\sigma_{spec} \xrightarrow{\tau} R_2\sigma_{spec}$. We set:

$$W' = (snd, srv, rcv, X, E, \sigma, \sigma_{spec}, R_2).$$

With this definition, $P' \equiv P^{W'}$ and $Q \rightarrow Q^{W'}$; moreover, W' is a possible world.

This concludes the proof of the authenticity property, Proposition 18. ▌

Proposition 18 is rather strong, so we obtain the secrecy property as a corollary (Proposition 19). This strength is convenient but not essential: weaker formulations of authenticity that do not imply secrecy would be satisfactory.

PROPOSITION 19. *If each pair* $(I_1, J_1), ..., (I_m, J_m)$ *is indistinguishable, then*

$$Sys(I_1, ..., I_m) \simeq Sys(J_1, ..., J_m).$$

Proof. When $I = (i, j, M)$ and $J = (i, j, M')$, the pair (I, J) is indistinguishable only if $F(\underline{i}, \underline{j}, M) \simeq F(\underline{i}, \underline{j}, M')$. Using the fact that testing equivalence is a congruence (Proposition 1), we obtain:

$$\begin{aligned} A1_{spec}(I, F) &= (vp)(A1(i, j, p) \mid p(x).F(\underline{i}, \underline{j}, M)) \\ &\simeq (vp)(A1(i, j, p) \mid p(x).F(\underline{i}, \underline{j}, M')) \\ &= A1_{spec}(J, F) \end{aligned}$$

If each pair $(I_1, J_1), ..., (I_m, J_m)$ is indistinguishable, then Propositions 1 and 18 permit the following calculation:

$$\begin{aligned} Sys(I_1, ..., I_m) &\simeq Sys_{spec}(I_1, ..., I_m) \\ &= (vK_{iS}{}^{i \in Prn})(vK_{Sj}{}^{j \in Prn}) \\ &\qquad \left(\prod_{k \in Ins} A1_{spec}(I_k, F) \;\middle|\; !S1 \;\middle|\; \prod_{j \in Prn} !B1(j, F_{spec}) \right) \\ &\simeq (vK_{iS}{}^{i \in Prn})(vK_{Sj}{}^{j \in Prn}) \\ &\qquad \left(\prod_{k \in Ins} A1_{spec}(J_k, F) \;\middle|\; !S1 \;\middle|\; \prod_{j \in Prn} !B1(j, F_{spec}) \right) \\ &= Sys_{spec}(J_1, ..., J_m) \\ &\simeq Sys(J_1, ..., J_m) \end{aligned}$$

This completes the proof of the secrecy property. ∎

7. FURTHER CRYPTOGRAPHIC PRIMITIVES

Although so far we have discussed only shared-key cryptography, other kinds of cryptography are also easy to treat within the spi calculus. In this section we show how to handle cryptographic hashing, public-key encryption, and digital signatures. We add syntax for these operations to the spi calculus and give their semantics. We thus provide evidence that our ideas are applicable to a wide range of security protocols, beyond those that rely on shared-key encryption. We believe that we may be able to deal similarly with Diffie–Hellman techniques and with secret sharing. However, protocols for oblivious transfer and for zero-knowledge proofs, for example, are probably beyond the scope of our approach.

7.1. Hashing

A cryptographic hash function has the properties that it is very expensive to recover an input from its image or to find two inputs with the same image.

Functions such as SHA and RIPE-MD are generally believed to have these properties [Sch96b].

When we represent hash functions in the spi calculus, we pretend that operations that are very expensive are altogether impossible. We simply add a construct to the syntax of terms of the spi calculus:

$L, M, N ::=$	terms
$\ldots$	as in Section 3.1
$H(M)$	hashing

The syntax of processes is unchanged. Intuitively, $H(M)$ represents the hash of M. The absence of a construct for recovering M from $H(M)$ corresponds to the assumption that H cannot be inverted. The lack of any equations $H(M) = H(M')$ corresponds to the assumption that H is free of collisions.

7.2. *Public-Key Encryption and Digital Signatures*

Traditional public-key encryption systems are based on key pairs. Normally, one of the keys in each pair is private to one principal, while the other key is public. Any principal can encrypt a message using the public key; only a principal that has the private key can then decrypt the message.

We assume that neither key can be recovered from the other. We could just as easily deal with the case where the public key can be derived from the private one. Much as in Section 3.1, we also assume that the only way to decrypt an encrypted packet is to know the corresponding private key; that an encrypted packet does not reveal the public key that was used to encrypt it; and that there is sufficient redundancy in messages so that the decryption algorithm can detect whether a ciphertext was encrypted with the expected public key.

We arrive at the following syntax for the spi calculus with public-key encryption. (This syntax is concise, rather than memorable.)

$L, M, N ::=$	terms
$\ldots$	as in Section 3.1
M^+	public part
M^-	private part
$\{\!\lbrack M \rbrack\!\}_N$	public-key encryption
$P, Q ::=$	processes
$\ldots$	as in Section 3.1
case L *of* $\{\!\lbrack x \rbrack\!\}_N$ *in* P	decryption

If M represents a key pair, then M^+ represents its public half and M^- represents its private half. Given a public key N, the term $\{\!\lbrack M \rbrack\!\}_N$ represents the result of the public-key encryption of M with N. In *case* L *of* $\{\!\lbrack x \rbrack\!\}_N$ *in* P, the variable x is bound in P. This construct is useful when N is a private key K^-; then it binds x to the M such that $\{\!\lbrack M \rbrack\!\}_{K^+}$ is L, if such an M exists.

It is also common to use key pairs for digital signatures. Private keys are used for signing, while public keys are used for checking signatures. We can represent digital signatures through the following extended syntax:

$L, M, N ::=$	terms
$\ldots$	as above
$[\{M\}]_N$	private-key signature
$P, Q ::=$	processes
$\ldots$	as in
case N *of* $[\{x\}]_M$ *in* P	decryption.

Given a private key N, the term $[\{M\}]_N$ represents the result of the signature of M with N. Again, the variable x is bound in P in the construct *case* N *of* $[\{x\}]_M$ *in* P. This construct is dual to *case* L *of* $\{[x]\}_N$ *in* P. The new construct is useful when N is a public key K^+; then it binds x to the M such that $[\{M\}]_{K^-}$ is L, if such an M exists. (Thus, we are assuming that M can be recovered from the result of signing it; but there is no difficulty in dropping this assumption.)

Formally, the semantics of the new constructs is captured with two new rules for the reduction relation:

(Red Public Decrypt) *case* $\{[M]\}_{N^+}$ *of* $\{[x]\}_{N^-}$ *in* $P > P[M/x]$

(Red Signature Check) *case* $[\{M\}]_{N^-}$ *of* $[\{x\}]_{N^+}$ *in* $P > P[M/x]$

We believe that our basic theoretical results for the spi calculus still apply.

As a small example, we can write the following public-key analogue for the protocol of Section 3.2.1:

$$
\begin{aligned}
A(M) &\triangleq \overline{c_{AB}}\langle \{[M, [\{H(M)\}]_{K_A^-}]\}_{K_B^+} \rangle \\
B &\triangleq c_{AB}(x).\, \mathit{case}\ x\ \mathit{of}\ \{[y]\}_{K_B^-}\ \mathit{in} \\
&\quad \mathit{let}\ (y_1, y_2) = y\ \mathit{in} \\
&\quad \mathit{case}\ y_2\ \mathit{of}\ [\{z\}]_{K_A^+}\ \mathit{in} \\
&\quad [H(y_1)\ \mathit{is}\ z]\ F(y_1) \\
\mathit{Inst}(M) &\triangleq (\nu K_A)(\nu K_B)(A(M) \mid B).
\end{aligned}
$$

In this protocol, A sends M on the channel c_{AB}, signed with A's private key and encrypted under B's public key; the signature is applied to a hash of M rather than to M itself. On receipt of a message on c_{AB}, B decrypts using its private key, checks A's signature using A's public key, checks the hash, and applies F to the body of the message (to M). The key pairs K_A and K_B are restricted; but there would be no harm in sending their public parts K_A^+ and K_B^+ on a public channel.

Undoubtedly, other formalizations of public-key cryptography are possible, perhaps even desirable. In particular, we have represented cryptographic operations at an abstract level and do not attempt to model closely the properties of any one algorithm. We are concerned with public-key encryption and digital signatures in general rather than with their RSA implementations, say. The RSA system satisfies equations that our formalization does not capture. For example, in the RSA system, $[\{\{[M]\}_{K^+}\}]_{K^-}$ equals M. We leave the treatment of those equations for future work.

8. CONCLUSIONS

We have applied both the standard pi calculus and the new spi calculus in the description and analysis of security protocols. As examples, we chose protocols of the sort commonly found in the authentication literature. We showed how to represent the protocols, how to express their security properties, and how to prove some of these properties. Our model of protocols takes into account the possibility of attacks, but does not require writing explicit specifications for an attacker. In particular, we express secrecy properties as simple equations that mean indistinguishability from the point of view of an arbitrary attacker. To our knowledge, this sharp treatment of attacks has not been previously possible.

Although our examples are small, we have found them instructive. Some of the techniques that we developed may be amenable to automation; the experience in other process algebras is encouraging. Moreover, there seems to be no fundamental difficulty in writing other kinds of examples, such as protocols for electronic commerce. Unfortunately, the specifications for those protocols do not yet seem to be fully understood, even in informal terms [Mao96].

In both the pi calculus and the spi calculus, restriction and scope extrusion play a central role. The pi calculus provides an abstract treatment of channels, while the spi calculus expresses the cryptographic operations that usually underlie channels in systems for distributed security. Thus, the pi calculus and the spi calculus are appropriate at different levels.

Those two levels are however related. In particular, as we have discussed briefly, we can specify a security protocol abstractly and then implement it using cryptography. Similarly, we may give an API (application programming interface) for secure channels and implement it on top of an API for cryptography. In more formal terms, it should be possible to define cryptographic implementations for the pi calculus, translating restricted channels into public channels with encryption. Implementation relations such as these are useful in practice; they seem worth studying further.

APPENDICES

The Appendices contain proofs for claims that appear without proof in the main body of this paper. Proofs for auxiliary results can be found in a technical report [AG97a].

A. Proofs about Commitment

In this section we prove Propositions 2, 3, and 4 from Section 5.1.

We begin with a lemma that relates the free names of a process to the free names of any agent to which it commits.

LEMMA 20. (1) *If* $P \xrightarrow{\tau} \tau Q$ *then* $fn(Q) \subseteq fn(P)$.

(2) *If* $P \xrightarrow{m} (x)Q$ *then* $\{m\} \cup fn(Q) \subseteq fn(P)$.

(3) *If* $P \xrightarrow{\bar{m}} (v\vec{n})\langle M\rangle\, Q$ *then* $\{m\} \cup fn((v\vec{n})\langle M\rangle\, Q) \subseteq fn(P)$ *and* $\{\vec{n}\} \subseteq fn(M)$.

The purpose of the next lemma is to show that $P \xrightarrow{\tau} Q$ implies $P \to Q$, half of Proposition 3.

LEMMA 21. (1) If $P \xrightarrow{m} (x)\, Q$ *then there are* Q_1, Q_2, *and names* $\vec{p}$ *such that* $m \notin \{\vec{p}\}$, $P \equiv (v\vec{p})(Q_1 \mid m(x).Q_2)$, *and* $Q[M/x] \equiv ((v\vec{p})(Q_1 \mid Q_2))[M/x]$ *for any closed* M.

(2) *If* $P \xrightarrow{\bar{m}} (v\vec{n})\langle M\rangle\, Q$ *then there are* Q_1, Q_2, *and names* $\vec{p}$ *such that* $m \notin \{\vec{p}\}$, $fn(M) \cap \{\vec{p}\} = \varnothing$, $P \equiv (v\vec{n}, \vec{p})(Q_1 \mid \bar{m}\langle M\rangle.Q_2)$, *and* $Q \equiv (v\vec{p})(Q_1 \mid Q_2)$.

(3) *If* $P \xrightarrow{\tau} Q$ *then* $P \to Q$.

The key fact we need for the other direction of Proposition 3 is that structural equivalence is a strong bisimulation.

LEMMA 22. $P \equiv Q$ *implies that*:

(1) *whenever* $P \xrightarrow{\alpha} A$ *there is* B *with* $Q \xrightarrow{\alpha} B$ *and* $A \equiv B$;

(2) *whenever* $Q \xrightarrow{\alpha} B$ *there is* A *with* $P \xrightarrow{\alpha} A$ *and* $A \equiv B$.

Hence structural equivalence is a strong bisimulation.

We can now prove the three propositions claimed in Section 5.1.

Proof of Proposition 2

$P \downarrow \beta$ iff $\exists A(P \xrightarrow{\beta} A)$.

Proof. This is not entirely trivial, as the $\downarrow$ relation is defined using structural equivalence, but the transition relation $\xrightarrow{\beta}$ is not. We can easily show that $P \xrightarrow{\beta} A$ implies $P \downarrow \beta$ by induction on the derivation of $P \xrightarrow{\beta} A$, using (Barb Struct) where necessary. On the other hand, we can show that $P \downarrow \beta$ implies $\exists A(P \xrightarrow{\beta} A)$ by induction on the derivation of $P \downarrow \beta$. The case of (Barb Struct) needs the fact that if $\exists A(P \xrightarrow{\beta} A)$ and $P \equiv Q$ then $\exists A(Q \xrightarrow{\beta} A)$ also, which follows from Lemma 22. ▌

Proof of Proposition 3

$P \to Q$ iff $P \xrightarrow{\tau} \equiv Q$.

Proof. For the backwards direction suppose $P \xrightarrow{\tau} R$ and $R \equiv Q$. By Lemma 21(3), $P \to R$, and then $P \to Q$ by (React Struct).

We can show that $P \to Q$ implies that there exists R such that $P \xrightarrow{\tau} R$ and $R \equiv Q$ by induction on the derivation of $P \to Q$. The only interesting case is (React Struct). Suppose that $P \to Q$ follows from $P \equiv P'$, $P' \to Q'$, and $Q' \equiv Q$. By induction hypothesis, $P' \xrightarrow{\tau} Q''$ with $Q'' \equiv Q'$. By Lemma 22, structural equivalence is a strong bisimulation, so $P \xrightarrow{\tau} R$ for some R such that $R \equiv Q''$. This with the previous equations gives $R \equiv Q$ as required. ▌

Proof of Proposition 4

P passes a test (R, β) iff there exist an agent A and a process Q such that $P \mid R \xrightarrow{\tau}{}^* Q$ and $Q \xrightarrow{\beta} A$.

Proof. By definition, P passes a test (R, β) iff $P \mid R \Downarrow \beta$, which holds iff there is Q with $P \mid R \to^* Q$ and $Q \downarrow \beta$, which by (Barb Struct), Lemma 22, and Propositions 2 and 3 is equivalent to there being Q and A with $P \mid R \xrightarrow{\tau}{}^* Q$ and $Q \xrightarrow{\beta} A$. ▌

B. Proofs about Replication

This section is devoted to lemmas concerning the interaction between replication and commitment, reaction, and convergence.

LEMMA 23. (1) *If* $!P \xrightarrow{m} (x)\, Q$, *then there is* R *with* $P \xrightarrow{m} (x)\, R$ *and* $Q[M/x] \equiv R[M/x] \mid !P$ *for any closed* M.

(2) *If* $!P \xrightarrow{\vec{m}} (v\vec{n})\langle M \rangle\, Q$, *then there is* R *with* $P \xrightarrow{\vec{m}} (v\vec{n})\langle M \rangle\, R$ *and* $Q \equiv R \mid !P$.

(3) *If* $!P \xrightarrow{\tau} Q$, *then there is* R *with* $P \mid P \xrightarrow{\tau} R$ *and* $Q \equiv R \mid !P$.

Intuitively, part (3) states that any reaction of $!P$ can be obtained from two copies of P running in parallel. As Pierce and Sangiorgi [PS96] have remarked, we can strengthen part (3) to require only one copy of P, but this stronger property would fail for an extended language with a choice construct. The claim with two copies would remain true for such an extended language.

LEMMA 24. *Suppose* $!P \mid R \xrightarrow{\tau} Q$. *Then there is* Q' *such that* $Q \equiv !P \mid Q'$ *and* $P \mid P \mid R \xrightarrow{\tau} Q'$.

For $n \geqslant 0$, we let $P \to^n Q$ mean that $P = P_0 \to P_1 \to P_2 \to \cdots \to P_n = Q$ for some processes $P_0, P_1, \ldots, P_n$.

LEMMA 25. (1) *Whenever* $!P \mid R \to^n Q$ *there is* Q' *with* $(\prod_{i \in 1..2n} P) \mid R \to^n Q'$ *and* $Q \equiv !P \mid Q'$.

(2) *Whenever* $!P \mid R \Downarrow \beta$ *there is* n *such that* $(\prod_{i \in 1..n} P) \mid R \Downarrow \beta$.

PROPOSITION 26. *If* $(v\vec{p})(P_1 \mid \prod_{i \in 1..n} P_2) \simeq (v\vec{p})(Q_1 \mid \prod_{i \in 1..n} Q_2)$ *for all* $n \geqslant 0$, *then* $(v\vec{p})(P_1 \mid !P_2) \simeq (v\vec{p})(Q_1 \mid !Q_2)$.

PROPOSITION 27. $!(P \mid Q) \simeq !P \mid !Q$.

C. Proofs about Equivalences

C.1. Testing Equivalence

The following are auxiliary facts needed for the proofs in this section.

LEMMA 28. (1) *$P \Downarrow \beta$ and $P \equiv Q$ imply $Q \Downarrow \beta$.*

(2) *$P \Downarrow \beta$ implies $P \mid Q \Downarrow \beta$.*

(3) *If $(\nu m)\, P \xrightarrow{\tau} R$ there is Q with $P \xrightarrow{\tau} Q$ and $R = (\nu m)\, Q$.*

(4) *If $(\nu m)\, P \downarrow \beta$ then $P \downarrow \beta$ and $\beta \notin \{m, m\}$.*

(5) *$(\nu m)\, P \Downarrow \beta$ iff $P \Downarrow \beta$ and $\beta \notin \{m, m\}$.*

LEMMA 29. $\equiv \subseteq \simeq$.

If $\mathscr{R}$ is a relation on closed processes, we let its open extension $\mathscr{R}^\circ$ be the relation on arbitrary processes such that $P\, \mathscr{R}^\circ Q$ if and only if $P\sigma\, \mathscr{R}\, Q\sigma$ for any substitution σ of closed terms for variables such that both $P\sigma$ and $Q\sigma$ are closed.

A congruence on closed processes is an equivalence relation $\mathscr{S}$ on closed processes such that $P\, \mathscr{S}\, Q$ implies $\mathscr{C}[P]\, \mathscr{S}\, \mathscr{C}[Q]$ for every closed context $\mathscr{C}$. Similarly, a congruence on open processes is an equivalence relation $\mathscr{S}$ on open processes such that $P\, \mathscr{S}\, Q$ implies $\mathscr{C}[P]\, \mathscr{S}\, \mathscr{C}[Q]$ for every context $\mathscr{C}$. The notion of precongruence is analogous, except that a precongruence must be a preorder instead of an equivalence relation.

We give an alternative characterization of congruence and precongruence that avoids the use of contexts. When $\mathscr{R}$ is a relation on open processes, we let its *compatible refinement* $\hat{\mathscr{R}}$ be the relation on open processes given by the rules in Fig. 3.

$$\text{(Comp Out)}\quad \frac{P\, \mathcal{R}\, Q}{\overline{M}\langle N\rangle.P\, \widehat{\mathcal{R}}\, \overline{M}\langle N\rangle.Q} \qquad \text{(Comp In)}\quad \frac{P\, \mathcal{R}\, Q}{M(x).P\, \widehat{\mathcal{R}}\, M(x).Q}$$

$$\text{(Comp Par)}\quad \frac{P_1\, \mathcal{R}\, Q_1 \quad P_2\, \mathcal{R}\, Q_2}{P_1 \mid P_2\, \widehat{\mathcal{R}}\, Q_1 \mid Q_2} \qquad \text{(Comp Res)}\quad \frac{P\, \mathcal{R}\, Q}{(\nu n)P\, \widehat{\mathcal{R}}\, (\nu n)Q} \qquad \text{(Comp Repl)}\quad \frac{P\, \mathcal{R}\, Q}{!P\, \widehat{\mathcal{R}}\, !Q}$$

$$\text{(Comp Match)}\quad \frac{P\, \mathcal{R}\, Q}{[M\ is\ N]\, P\, \widehat{\mathcal{R}}\, [M\ is\ N]\, Q} \qquad \text{(Comp Nil)}\quad \frac{}{\mathbf{0}\, \widehat{\mathcal{R}}\, \mathbf{0}}$$

$$\text{(Comp Split)}\quad \frac{P\, \mathcal{R}\, Q}{let\ (x,y) = M\ in\ P\, \widehat{\mathcal{R}}\, let\ (x,y) = M\ in\ Q}$$

$$\text{(Comp IntCase)}\quad \frac{P_1\, \mathcal{R}\, Q_1 \quad P_2\, \mathcal{R}\, Q_2}{case\ M\ of\ 0 : P_1\ suc(x) : P_2\, \widehat{\mathcal{R}}\, case\ M\ of\ 0 : Q_1\ suc(x) : Q_2}$$

$$\text{(Comp Decrypt)}\quad \frac{P\, \mathcal{R}\, Q}{case\ N\ of\ \{x\}_M\ in\ P\, \widehat{\mathcal{R}}\, case\ N\ of\ \{x\}_M\ in\ Q}$$

FIG. 3. Rules of compatible refinement.

LEMMA 30. *Suppose that $\mathscr{R}$ is a preorder. Then $\mathscr{R}$ is a precongruence (closed under arbitrary contexts) iff $\hat{\mathscr{R}} \subseteq \mathscr{R}$.*

See [Gor95] for the proof of a similar proposition.

LEMMA 31. *The open extension of testing equivalence, $\simeq^\circ$, is a congruence.*

We obtain:

Proof of Proposition 1. (1) Structural equivalence implies testing equivalence.

(2) Testing equivalence is reflexive, transitive, and symmetric.

(3) Testing equivalence is a congruence on closed processes.

Proof. That structural equivalence implies testing equivalence is said in Lemma 29. Whenever $\mathscr{S}$ is a relation on closed processes and $\mathscr{S}^\circ$ is a congruence on open processes, $\mathscr{S}$ is a congruence on closed processes. ▮

The remainder of this section concerns some testing equivalences that we use in reasoning about protocols.

PROPOSITION 32. *For any closed process P, $P \simeq \tau.P$.*

LEMMA 33. *For any P such that $fv(P) \subseteq \{x\}$ and any distinct names m and n, $m(x).(vn)\, P \simeq (vn)\, m(x).P$.*

LEMMA 34. *Let n be a name, M a (possibly open) term, $\{N_i \mid i \in I\}$ a set of distinct closed terms, and $\{P_i \mid i \in I\}$ a set of (possibly open) processes, where I is a finite set of indices. Then $\prod_{i \in I} [M \text{ is } N_i](vn)\, P_i \simeq^\circ (vn) \prod_{i \in I} [M \text{ is } N_i]\, P_i$.*

C.2. Barbed Equivalence

Proof of Proposition 5. (1) Barbed equivalence is reflexive, transitive, and symmetric.

(2) Strong bisimilarity implies barbed equivalence.

(3) Structural equivalence implies barbed equivalence.

(4) Barbed equivalence is preserved by restriction.

Proof. (1) As usual, we can show that the identity relation is a barbed bisimulation, that the composition of two barbed bisimulations yields a barbed bisimulation, and that the converse of a barbed bisimulation is a barbed bisimulation.

(2) It is enough to show that strong bisimilarity is a barbed bisimulation. Given Propositions 2 and 3 this is easy.

(3) By Lemma 22, structural equivalence is a strong bisimulation. By part (2), it is contained in barbed equivalence.

(4) It suffices to show that $\{((vn)\, P, (vn)\, Q) \mid P \mathrel{\dot{\sim}} Q\}$ is a barbed bisimulation. The proof is straightforward. ▮

Proof of Proposition 6. If $\mathscr{S}$ is a barbed bisimulation up to $\mathrel{\dot{\sim}}$ and restriction, then $\mathscr{S} \subseteq \mathrel{\dot{\sim}}$. A fortiori, if $\mathscr{S}$ is a barbed bisimulation up to $\mathrel{\dot{\sim}}$, then $\mathscr{S} \subseteq \mathrel{\dot{\sim}}$.

Proof. This proposition can be proved easily through the standard technique used in the pi calculus [MPW92]. (See [AG97a] for details.) An alternative is to rely on the modular framework recently developed by Sangiorgi [San94]. ▮

C.3. Barbed Congruence

The main task of this section is to show $\widehat{\sim}^{\circ} \subseteq \sim^{\circ}$, from which it follows that $\sim^{\circ}$ is a congruence. The following is an adaptation of the proof by Pierce and Sangiorgi [PS96].

We begin with two lemmas concerning replication and commitment.

LEMMA 35. $\widehat{\sim}^{\circ} \subseteq \sim^{\circ}$.

Now we can prove the basic facts about barbed congruence claimed in Section 5.2.3.

Proof of Proposition 7. (1) Barbed congruence is reflexive, transitive, and symmetric.

(2) Barbed congruence is a congruence on closed processes.

(3) Structural equivalence implies barbed congruence.

(4) Strong bisimilarity implies barbed congruence.

(5) Barbed congruence implies testing equivalence.

Proof. (1) Since $\dot{\sim}$ is an equivalence relation, so is $\sim$.

(2) Lemma 35 yields that the open extension of barbed congruence, $\sim^{\circ}$, is a congruence on open processes. It follows that barbed congruence is a congruence on closed processes.

(3) This follows from part (4), since we know from Lemma 22 that structural equivalence implies strong bisimilarity.

(4) It suffices to check that the following relation is a barbed bisimulation:

$$\mathscr{S} = \{(P \mid R, Q \mid R) \mid P \text{ and } Q \text{ strongly bisimilar}\}$$

We omit the routine proof, which involves using the commitment relation to analyze the possible barbs and reactions of $P \mid R$ and $Q \mid R$, and showing that they match up to $\mathscr{S}$.

(5) Suppose that $P \sim Q$, and consider any test (R, β). By definition of barbed congruence, $(P \mid R) \dot{\sim} (Q \mid R)$. Hence, $(P \mid R) \Downarrow \beta$ implies $(Q \mid R) \Downarrow \beta$ too. Therefore, $P \simeq Q$. ▮

D. Proofs about Underpinning

LEMMA 36. *Suppose* $E \vdash M$, $E \vdash N$, *and* $E \vdash \sigma$. *If* σ *is injective, then* $M\sigma = N\sigma$ *implies* $M = N$.

Proof of Lemma 9. Suppose that $E \vdash P$ and $E \vdash \sigma$, and that σ is injective.

(1) If $P\sigma > Q'$ then there is a process Q with $E \vdash Q$, $fv(Q) \subseteq fv(P)$, $fn(Q) \subseteq fn(P)$, and $Q' = Q\sigma$ such that, whenever $E \vdash \sigma'$ and σ' is injective, $P\sigma' > Q\sigma'$.

(2) If $P\sigma \xrightarrow{\alpha} A'$ then there is an agent A with $E \vdash A$, $fv(A) \subseteq fv(P)$, $fn(A) \subseteq fn(P)$, and $A' = A\sigma$ such that, whenever $E \vdash \sigma'$ and σ' is injective, $P\sigma' \xrightarrow{\alpha} A\sigma'$.

Proof. (1) By analysis of the rules that may yield $P\sigma > Q'$.

(Red Decrypt) Here $P = case\ M\ of\ \{x\}_N\ in\ R$ with $M\sigma = \{M'_1\}_{N\sigma}$ and $Q' = R\sigma[M'_1/x]$, given that we may assume that bound variable x is not in the domain or range of σ. Since $M\sigma = \{M'_1\}_{N\sigma}$, either M is a variable $y \in dom(\sigma)$ or a ciphertext $\{M_1\}_{M_2}$.
In the former case, $y\sigma = \{M'_1\}_{N\sigma}$ so $N\sigma$ must be a member of $keys(E)$, and therefore is a name, say K. Since the range of σ consists of ciphertexts, N itself must be the name K. But then we have $K \in keys(E)$ while also $K \in fn(P)$, which contradicts our assumption that $E \vdash P$.
Therefore $M = \{M_1\}_{M_2}$. It follows that $M_1\sigma = M'_1$ and $M_2\sigma = N\sigma$. By Lemma 36, $M_2 = N$. Let $Q = R[M_1/x]$. From $E \vdash P$ it follows that $E \vdash Q$ too. Further, $fv(Q) \subseteq fv(M_1) \cup (fv(R) - \{x\}) \subseteq fv(P)$ and $fn(Q) \subseteq fn(M_1) \cup fn(R) \subseteq fn(P)$. For any injective σ' with $E \vdash \sigma'$, we have:

$$\begin{aligned} P\sigma' &= case\ \{M_1\sigma'\}_{N\sigma'}\ of\ \{x\}_{N\sigma'}\ in\ R\sigma' \\ &> R\sigma'[M_1\sigma'/x] \\ &= (R[M_1/x])\,\sigma' \end{aligned}$$

So we have $P\sigma' > Q\sigma'$ as required.

(Red Match) Here $P = [N_1\ is\ N_2]\ Q$ with $N_1\sigma = N_2\sigma$ and $Q' = Q\sigma$. By Lemma 36, $N_1 = N_2$. From $E \vdash P$ it follows that $E \vdash Q$ too. Since Q is a part of P, $fv(Q) \subseteq fv(P)$ and $fn(Q) \subseteq fn(P)$. For any injective σ' with $E \vdash \sigma'$, we have $P\sigma' = [N_1\sigma'\ is\ N_2\sigma']\ Q\sigma' > Q\sigma'$ as required.

The other cases are routine, given that M must be a ciphertext if it is in the range of σ.

(2) By induction on the derivation of $P\sigma \xrightarrow{\alpha} A'$.

(Comm In) Here $P = M(x).Q$ with $M\sigma = m = \alpha$ and $A' = (x)(Q\sigma)$, where we may assume that bound variable x is not in the domain or range of σ. Since $M\sigma$ is a name, m, it must be that M itself is the name, since only ciphertexts are in the range of σ. Let $A = (x)Q$. From $E \vdash P$ it follows that $E \vdash A$ too. Further, $fv(A) = fv(Q) - \{x\} \subseteq fv(P)$ and $fn(A) = fn(Q) \subseteq fn(P)$. We have $A' = (x)(Q\sigma) = A\sigma$. For any injective σ' with $E \vdash \sigma'$, we have:

$$P\sigma' = m(x).Q\sigma' \xrightarrow{m} (x)(Q\sigma') = A\sigma'$$

as required.

(Comm Inter 1) Here $P = P_1 \mid P_2$, with $P_1\sigma \xrightarrow{m} F'$ and $P_2\sigma \xrightarrow{\bar{m}} C'$, $\alpha = \tau$, and $A' = F'dC'$. By induction hypothesis, there is F such that $F' = F\sigma$, $E \vdash F$, $fv(F) \subseteq fv(P_1)$, $fn(F) \subseteq fn(P_1)$, and $P_1\sigma' \xrightarrow{m} F\sigma'$ for all injective σ' with $E \vdash \sigma'$. By induction hypothesis, there is C such that $C' = C\sigma$, $E \vdash C$, $fv(C) \subseteq fv(P_2)$, $fn(C) \subseteq fn(P_2)$, and $P_2\sigma' \xrightarrow{\bar{m}} C\sigma'$ for all injective σ' with $E \vdash \sigma'$. Let $A = F@C$. Interaction, $@$, is defined so that it commutes with substitution, so we have $A\sigma = F\sigma@C\sigma = F'@C' = A'$. From $E \vdash F$ and $E \vdash C$ follows $E \vdash A$. Further, $fv(A) \subseteq fv(F) \cup fv(C) \subseteq fv(P_1) \cup fv(P_2) = fv(P)$ and $fn(A) = fn(F) \cup fn(C) \subseteq fn(P_1) \cup fn(P_2) = fn(P)$. For any injective σ' with $E \vdash \sigma'$, we have

$$\begin{aligned} P\sigma' &= P_1\sigma' \mid P_2\sigma' \\ &\xrightarrow{\tau} F\sigma'@C\sigma' \\ &= (F@C)\,\sigma' \end{aligned}$$

where the τ commitment follows using (Comm Inter 1) and the facts that $P_1\sigma' \xrightarrow{m} F\sigma'$ and $P_2\sigma' \xrightarrow{\bar{m}} C\sigma'$. We have obtained $P\sigma' \xrightarrow{\tau} A\sigma'$, as required.

(Comm Red) Here $P\sigma > Q'$ and $Q' \xrightarrow{\alpha} A'$. By part (1), there is Q with $E \vdash Q$, $fv(Q) \subseteq fv(P)$, $fn(Q) \subseteq fn(P)$, $Q' = Q\sigma$, and $P\sigma' > Q\sigma'$ for all injective σ' with $E \vdash \sigma'$. Since $E \vdash Q$ and $Q\sigma \xrightarrow{\alpha} A'$, by induction hypothesis, there is A with $E \vdash A$, $fv(A) \subseteq fv(Q)$, $fn(A) \subseteq fn(Q)$, $A' = A\sigma$, and $Q\sigma' \xrightarrow{\alpha} A\sigma'$ for all such σ'. By transitivity, we have $fv(A) \subseteq fv(P)$ and $fn(A) \subseteq fn(P)$. Further, for any injective σ' with $E \vdash \sigma'$, we have obtained $P\sigma' > Q\sigma'$ and $Q\sigma' \xrightarrow{\alpha} A\sigma'$, so by (Comm Red) $P\sigma \xrightarrow{\alpha} A\sigma'$, as required.

The case for (Comm Out) is similar to that for (Comm In). The case for (Comm Inter 2) is like that for (Comm Inter 1). Those for (Comm Par 1), (Comm Par 2), and (Comm Res) are by simple uses of the induction hypothesis. ▮

This lemma would still hold in a spi calculus with the mismatch operator mentioned in Section 4.2. The case for mismatch in part (1) would be like that of (Red Match), with a similar appeal to Lemma 36.

ACKNOWLEDGMENTS

Peter Sewell, Phil Wadler, and anonymous referees suggested improvements to a draft of this paper. Cynthia Hibbard provided editorial help. For most of the time we worked on this paper, Gordon held a University Research Fellowship awarded by the Royal Society.

Received May 8, 1997; final manuscript received March 22, 1998

REFERENCES

[Aba97] Abadi, M. (1997), Secrecy by typing in security protocols, *in* "Theoretical Aspects of Computer Software," Lecture Notes in Computer Science, Vol. 1281, pp. 611–638, Springer-Verlag, Berlin/New York.

[ABLP93] Abadi, M., Burrows, M., Lampson, B., and Plotkin, G. (1993), A calculus for access control in distributed systems, *ACM Trans. Program. Languages Systems* **15**(4), 706–734.

[AG97a] Abadi, M. and Gordon, A. D. (1997a), A calculus for cryptographic protocols: The Spi Calculus, Technical Report 414, University of Cambridge Computer Laboratory, January. [A revised version appeared as Digital Equipment Corporation Systems Research Center Report No. 149, January 1998]

[AG97b] Abadi, M., and Gordon, A. D. (1997b), A calculus for cryptographic protocols: The spi calculus, *in* "Proceedings of the Fourth ACM Conference on Computer and Communications Security," pp. 36–47.

[AG97c] Abadi, M., and Gordon, A. D. (1997), Reasoning about cryptographic protocols in the spi calculus, *in* "CONCUR'97: Concurrency Theory," Lecture Notes in Computer Science, Vol. 1243, pp. 59–73, Springer-Verlag, Berlin/New York.

[AG98] Abadi, M., and Gordon, A. D. (1998), A bisimulation method for cryptographic protocols, *in* "Proceedings of ESOP'98," Lecture Notes in Computer Science, pp. 12–26, Springer-Verlag, Berlin/New York.

[AN96] Abadi, M., and Needham, R. (1996), Prudent engineering practice for cryptographic protocols, *IEEE Trans. Software Engrg.* **22**(1), 6–15.

[BAN89] Burrows, M., Abadi, M., and Needham, R. M. (1989), A logic of authentication, *Proc. Roy. Soc. London Ser. A* **426**, 233–271. [A preliminary version appeared as Digital Equipment Corporation Systems Research Center Report No. 39, Feb. 1989]

[BB92] Berry, G., and Boudol, G. (1992), The chemical abstract machine, *Theoret. Comput. Sci.* **96**(1), 217–248.

[BN95] Boreale, M., and De Nicola, R. (1995), Testing equivalence for mobile processes, *Inform. and Comput.* **120**(2), 279–303.

[BR95] Bellare, M., and Rogaway, P. (1995), Provably secure session key distribution: The three party case, *in* "Proceedings of the 27th Annual ACM Symposium on Theory of Computing," pp. 57–66.

[DES77] Data encryption standard. Fed. Inform. Processing Standards Pub. 46, National Bureau of Standards, Washington DC, Jan. 1977.

[DH76] Diffie, W., and Hellman, M. (1976), New directions in cryptography, *IEEE Trans. Inform. Theory* **22**(6), 644–654.

[DH84] De Nicola, R., and Hennessy, M. C. B. (1984), Testing equivalences for processes, *Theoret. Comput. Sci.* **34**, 83–133.

[DY81] Dolev, D., and Yao, A. C. (1981), On the security of public key protocols, *in* "Proc. 22th IEEE Symposium on Foundations of Computer Science," pp. 350–357.

[GM95] Gray, J., and McLean, J. (1995), Using temporal logic to specify and verify cryptographic protocols (progress report), *in* "Proceedings of the 8th IEEE Computer Security Foundations Workshop," pp. 108–116.

[Gor95] Gordon, A. D. (1995), Bisimilarity as a theory of functional programming, Minicourse, BRICS Notes Series NS-95-3, BRICS, Aarhus University.

[Hoa85] Hoare, C. A. R. (1985), "Communicating Sequential Process," Prentice–Hall International, Englewood Cliffs, NJ.

[Kem89] Kemmerer, R. A. (1989), Analyzing encryption protocols using formal verification techniques, *IEEE J. Selected Areas Commun.* **7**, 448–457.

[LABW92] Lampson, B., Abadi, M., Burrows, M., and Wobber, E. (1992), Authentication in distributed systems: Theory and practice, *ACM Trans. Comput. Systems* **10**(4), 265–310.

[Lie93] Liebl, A. (1993), Authentication in distributed systems: A bibliography, *ACM Oper. Systems Rev.* **27**(4), 31–41.

[Low96] Lowe, G. (1996), Breaking and fixing the Needham–Schroeder public-key protocol using FDR, *in* "Tools and Algorithms for the Construction and Analysis of Systems," Lecture Notes in Computer Science, Vol. 1055, pp. 147–166, Springer-Verlag, Berlin/New York.

[Mao96] Mao, W. (1996), On two proposals for on-line bankcard payments using open networks: Problems and solutions, *in* "IEEE Symposium on Security and Privacy," pp. 201–210.

[MCF87] Millen, J. K., Clark, S. C., and Freedman, S. B. (1987), The Interrogator: Protocol security analysis, *IEEE Trans. Software Engrg.* **13**(2), 274–288.

[Mea92] Meadows, C. (1992), Applying formal methods to the analysis of a key management protocol, *J. Comput. Security* **1**(1), 5–36.

[Mil89] Milner, R. (1989), "Communication and Concurrency," Prentice–Hall International, Englewood Cliffs, NJ.

[Mil92] Milner, R. (1992), Functions as processes, *Math. Structures Comput. Sci.* **2**, 119–141.

[Mil95a] Millen, J. K. (1995), The Interrogator model, *in* "IEEE Symposium on Security and Privacy," pp. 251–260.

[Mil95b] Milner, R. (1995), The π-calculus, undergraduate lecture notes, Cambridge University.

[MPW92] Milner, R., Parrow, J., and Walker, D. (1992), A calculus of mobile processes, Parts I and II, *Inform. and Comput.* 1–40; 41–77.

[MS92] Milner, R., and Sangiorgi, D. (1992), Barbed bisimulation, *in* "Proceedings of 19th International Colloquium on Automata, Languages and Programming (ICALP'92)," Lecture Notes in Computer Science, Vol. 623, Springer-Verlag, Berlin/New York.

[NS78] Needham, R. M., and Schroeder, M. D. (1978), Using encryption for authentication in large networks of computers, *Comm. ACM* **21**(12), 993–999.

[Pau97] Paulson, L. (1997), Proving properties of security protocols by induction, *in* "Proceedings of the 10th IEEE Computer Security Foundations Workshop," pp. 70–83.

[PS96] Pierce, B., and Sangiorgi, D. (1996), Typing and subtyping for mobile processes, *Math. Structures Comput. Sci.* **6**(5), 409–453.

[RSA78] Rivest, R. L., Shamir, A., and Adleman, L. (1978), A method for obtaining digital signatures and public-key cryptosystems, *Comm. ACM* **21**(2), 120–26.

[San94] Sangiorgi, D. (1994), On the bisimulation proof method, Technical Report ECS-LFCS-94-299, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, August.

[Sch96a] Schneider, S. (1996), Security properties and CSP, *in* "IEEE Symposium on Security and Privacy," pp. 174–187.

[Sch96b] Schneier, B. (1996), "Applied Cryptography: Protocols, Algorithms, and Source Code in *C*," second ed., Wiley, New York.